Telegram Listing

Telegrams for Configuring and Operating the NAV350 Laser Positioning Sensor

Navigating the route to improved productivity

SICK
Sensor Intelligence.

Telegram listing version

For the latest version of this telegram listing (PDF), see www.sick.com.

1 Introduction

1.1 Function

This document shows how to send telegrams via terminal program to the NAV 350. The focus is on ASCII (also in Hex) in SOPAS ET. Descriptions of the commands, the parameters and the expected response after sending a telegram with the compact command CoLa (**Co**mmand **La**nguage) are described. There are two types of the **Co**mmand **La**nguage: ASCII- (**A**) and Binary- (**B**) format. Parameters access as well as method calls are supported. The common framing and rules of the binary protocol are described in the Introduction as well but particular methods have to be converted from ASCII if CoLa B is used.

Attention: Some commands may change during SICK development processes. Please always use the latest version of the “Telegram Listing”.

1.2 Command Formats

1.2.1 Protocols in SOPAS ET

To connect to the telegram section in SOPAS ET choose:
Tools → Terminal → Connections → New Connection...
At this point you have the choice between using the default protocol of the connected device and creating a “user defined connection”.

Attention: If you want to communicate via telegram in SOPAS ET, close all open windows of the scanner in SOPAS ET; otherwise you will get continuous status information that will make it impossible to read your requests and the corresponding responses.

1.2.2 Binary Protocol

The binary protocol of the scanner has always a fixed length and the command string can be converted from the ASCII commands described in this document. The binary protocol has a special framing so that the scanner is able to recognize the start of a binary telegram. The string has to start with 4 STX symbols (i.e. 02 02 02 02) that is followed by the length of the telegram in HEX (i.e.: 00 00 00 1B).

Example:
Binary: 02 02 02 02 00 00 00 1B 73 4D 4E 20 53 65 74 41 63 63 65 73 73 4D 6F 64 65 20 33 46 34 37 32 34 37 34 34 72

Special Characters: Header: 02 02 02 02; Length: 00 00 00 1B; Space: 20; Checksum: 72

The length can be created by counting every letter of the command in ASCII or every character pair in binary (without checksum and framing but with blanks) and convert the sum into HEX. Zeros are added in front until a string of eight characters is built.
The command itself starts after the length characters. Every single letter of the written command is converted to HEX turning into a pair of numbers, followed by a blank and then the parameters also converted in HEX. In between parameters there are no blanks. The “Checksum” is built with XOR beginning the calculation right after the length (i.e. starting with 73 in the example above).

Attention: The response has the same structure as the request. Therefore the expected parameter is also followed by a checksum which may lead to confusion.

1.2.3 ASCII Protocol

The framing of a telegram in ASCII is a <STX> at the start and an <ETX> at the end of each telegram. Commands are written as letters, followed by the parameters as defined in this document. There has to be a blank in between the command and the parameters and also in between each parameter (as shown in the example below as _).

Example:
<STX>sMN_SetAccessMode_3_F4724744<ETX>

In HEX the command start with 02 and ends with 03. The spaces are marked as 20. Single numbers that are converted to HEX always get a 3 in front.

Example:
02734D4E20536574416363657373 4D6F64652033 20 46 34 37 32 34 37 34 34 03

(Both examples show the same command and parameters.)

1.2.3.1 Request (User to Device)

In the ASCII decimal the user may choose one of the following notations to represent the value of one parameter in a command:

- Decimal notation
  Values prefixed **with** "+" or "-" are interpreted as decimal value.
  Example: Value to be sent: 319 (= 3F hex) ➔ **ASCII:** '+',‘3‘,‘1‘, ‘9’

- Hexadecimal notation
  Values **without** a prefixed "+" or "-" are interpreted as a hexadecimal value
  Example: Value to be sent: 3F hex. (= 319 dec) ➔ **ASCII:** '3‘,‘F'

  The NAV350 interprets each parameter individually; therefore the different notations can be mixed within a command string.

Attention: In an ASCII hexadecimal telegram the first notation has to be used but the number has to be transformed to HEX (see example below).

Example:

If you want a value of 319 in a command written in HEX: 319 → in HEX: 2B 33 31 39

HEX: + → 2B ; - → 2D

1.2.3.2 Response (Device to User)

The scanner always responds in hexadecimal notation if no parameter (in decimal) was set before.

Attention: All values are returned from the scanner as HEX values, if they were not set before, so they have to be converted afterwards. Once a value is set in ASCII decimal it is also returned as ASCII decimal value. There are some methods (i.e. *mNAVGetTimestamp* (p.47)) that just request a value so the response is always in HEX and therefore it has to be converted afterwards. For these methods the corresponding range only indicates the valid range (in decimal) but not the return value.

1.2.4 Summary Example and Format Explanation

This paragraph shall clarify the differences between the three kinds of command formats.

At first the command basics used in the telegrams are summarized in the following table.

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 by name	sRN	73 52 4E	
Write by name	sWN	73 57 4E	
Method	sMN	73 4D 4E	
Space	{SPC}	20	20

Tab.1-1: Command basics

The following table shows the layout that is used to describe the commands in this document but in order to demonstrate the difference between the two protocols the “Values Binary” column is added.

Telegram structure: sMN SetAccessMode				
Command part	**Description**	**Type**	**Values ASCII**	**Values Binary**
Command Type	SOPAS by name	String	**sMN**	**73 4D 4E**
Command	User level	String	**SetAccessMode**	**53 65 74 41 63 63 65 73 73 4D 6F 64 65**
User level	Select user level	Int_8	**2 = maintenance 3 = authorized client**	**02 = maintenance 03 = authorized client**
Password	“Hash” - value for the user level “Maintenance” “Authorized Client”	UInt_32	**B21ACE26** **F4724744**	**B2 1A CE 26** **F4 72 47 44**

Tab.1-2: Telegram structure: sMN SetAccessMode

The last table shows a telegram sting in the three different command formats (as they can be seen if used in the SOPAS telegram).

Example String:

sMN SetAccessMode 03 F4724744	
ASCII	<STX>sMN{SPC}SetAccessMode{SPC}3{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
Binary	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

Tab.1-3: Example string

1.3 Data Types of Variables

The following data types are specified for the variable values in the telegram:

Type	Size (bits)	Range of Value (decimal)	Sign
Bool_1	8	0 or 1	unsigned
UInt_8	8	0 ... 255	unsigned
Int_8	8	–128 ... +127	signed
UInt_16	16	0 ... 65535	unsigned
Int_16	16	–32768 ... +32767	signed
UInt_32	32	0 ... 4294967295	unsigned
Int_32	32	–2147483648 ... +2147483647	signed
Float_32	32	$\pm$~$10^{-44,85}$ ... $+10^{38,53}$	signed
Enum_8	8	0...255	unsigned
Enum_16	16	0..65535	unsigned
String	Depends on content	Note: Strings not terminated by 0	

Tab.1-4: Data types of variables

Note

- The figures in the table column "Size (bits)" refer to the binary transmission of numerical parameters. The ASCII representation differs from these numbers depending on their notation as decimal or hexa-decimal values.

- The figures in the table column "Range of value (decimal)" refer to the mathematically possible Range/Value of values for the variables. The current range of values may differ and is described in the “Command” chapter for each individual parameter of a command.

1.4 Storable Parameter

The following table shows default settings for parameters and contents stored in the flash memory. More parameters are storable by the command „mEEwriteAll", if logged in as „Authorized client".

Parameter	Command	Range	Corresponding Value & Default
Serial configuration			
Baudrate	SIHstBaud	**6** **7** **8** **9**	**19200** **38400** **57600** **115200 (Default)**
Data bits	Data Bits	**7..8**	**7_data bits** **8_data bits (Default)**
Parity	Parity Check	**0..4**	**0 = none** **1 = odd** **2 = even (Default)** **3 = mark** **4 = space**
Stopbit	SIHstStop	**0** **1**	**1 (Default)** **2**
Device ID	NetDeviceID	**1 ... 127**	**(Default: 1)**
Ethernet configuration			
IP-Address	EtherIPAddress	**0 ... 255 (4x)**	**192.168.1.10 (Default)**
Gateway	EtherIPGateAddress	**0 ... 255 (4x)**	**0.0.0.0 (Default)**
Subnetmask	EtherIPMask	**0 ... 255 (4x)**	**255.255.255.0 (Default)**
Interfaced speed and duplex mode	EtherIPSpeedDuplex	**0** **1** **2** **3** **4**	**auto (Default)** **10MB half** **10MB full** **100MB half** **100MB full**
Addressmode (DHCP)	EtherAddressingMode	**0** **1**	**disabled (Default)** **enabled**

Measurement parameter			
Percent	NLMDReflThreshold	**0...100 in %**	**Default: 50%**
SlidingMean	NPOSSlidingMean	**1 ... 63**	**Default: 1**
Filter	NLMDLandmarkMatching	**0** **1** **2**	**optimal (Default)** **optimal + angle** **optimal + angle + partly covered**
WinLow	NCORIdentWindow	**100 ... 2000 mm**	**Default: 300**
WinHigh		**100 ... 2000 mm**	**Default: 300**
DistLow		**500 ... 70000 mm**	**Default: 500**
DistHigh		**500 ... 70000 mm**	**Default: 70000**
NClosest	NLMDnClosest	**2 ... 40**	**Default: 0**
RFr	NLMDActionRadius	**500 ... 70000 mm**	**Default: 500**
RTo		**500 ... 70000 mm**	**Default: 70000**
Sector0 /1 /2 /3	NLMDMutedSectors	**0 ... 359999 mdeg**	**inactive**
Direction	NEVACoordOrientation	**0** **1**	**clockwise** **counterclockwise (Default)**

Tab.1-5: Storable parameter by the command "mEEwriteAll", if logged in as "authorized client"

1.5 Initialization Sequence

This sequence will initialize the NAV350 and start the navigation mode (mode 4).
The values of step 3 and 4 are just recommended default values that have to be replaced to fulfill individual requirements.

	Step	Command	Page	Parameter	Values
1	Log-in	**sMN SetAccessMode**	*45*	*newMode password*	*3 F4724744*
2	Standby mode	**sMN mNEVAChangeState**	43	mode	**1**
3	Write current Layer	**sWN NEVACurrLayer**	20	*currLayer*	**7**
4	Set positioning data format	**sWN NPOSPoseDataFormat**	*27*	*outputMode*	**1 0**
5	Navigation mode	**sMN mNEVAChangeState**	43	mode	**4**
6	Requesting the position	**sMN mNPOSGetPose**	*72*	*wait*	**1**

Tab.1-6: Example for an initialization sequence

1.6 Sequence for Reflector Mapping

This sequence generates a valid layout in the memory of the scanner for navigation. In order to use this sequence you have to switch to MAPPING (mode 3) first.

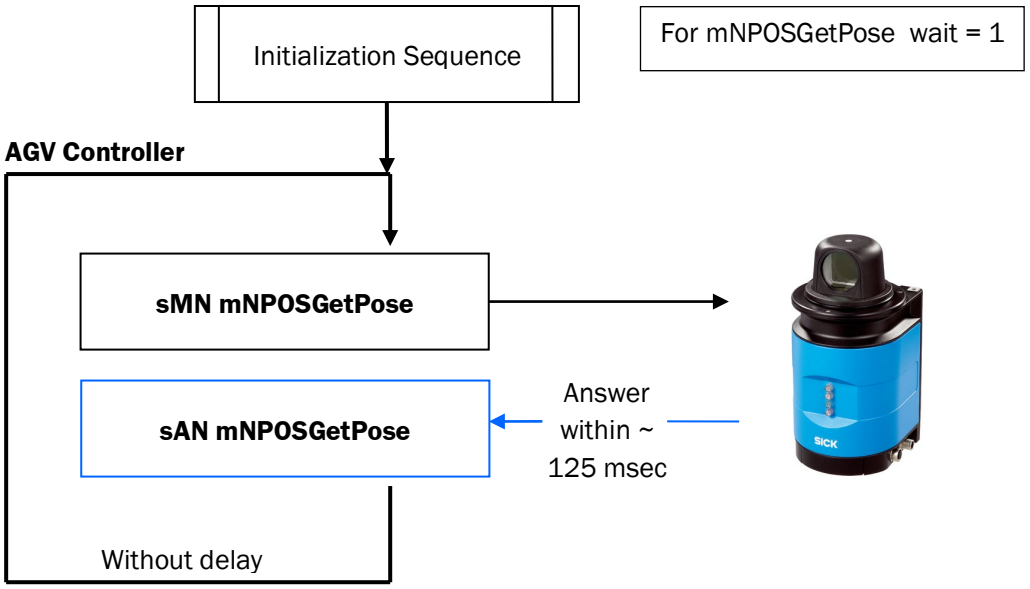
	Step	Command	Page	Parameter
1	Configure mapping (Settings for the mapping)	**sWN NMAPMapCfg**	*24*	*mean negative x y phi*
2	Select the layer number	**sWN NEVACurrLayer**	20	*Current layer*
2	Do mapping (Performs mapping and sends the measured reflectors data to PC/PLC)	**sMN mNMAPDoMapping**	70	
3	Add landmark (Writes 1..50 landmarks to the scanner)	**sMN mNLAYAddLandmark**	59	*landmarkData*
4	Store layout permanent (Saves the reflectors in the flash memory)	**sMN mNLAYStoreLayout**	*65*	

Tabelle 1-7: Sequence for reflector mapping

1.7 Principle of Navigation Mode

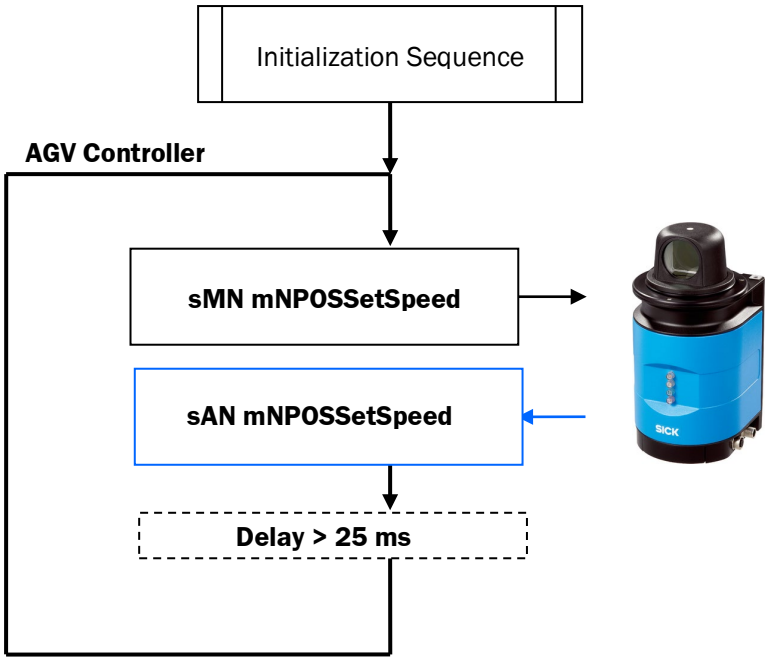
This chapter shows the sequences from the initialization to the navigational loop to request the current position from the scanner.

1.7.1 Navigational Loop



1.7.2 Providing Speed Loop

The providing speed loop runs parallel up to 40Hz to the navigational loop.



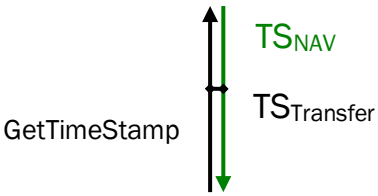
1.8 Timing Set Speed

1.8.1 Synchronization

The formula shows the resulting timing offset after requesting the scanner time between the timer of the AGV and the NAV350 and considers also the influence of the transmission time.

GetTimeStamp →

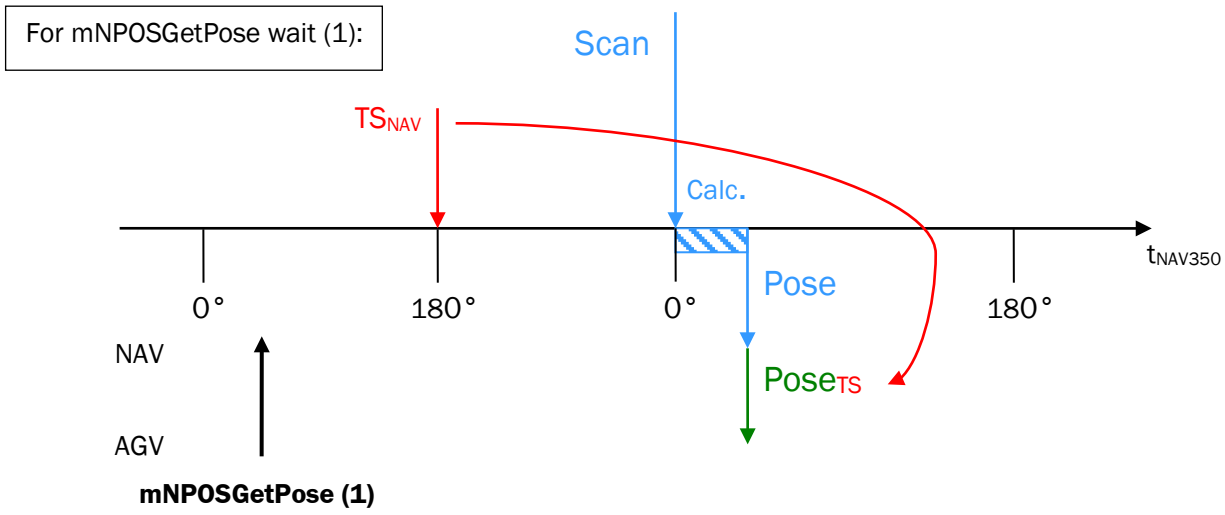
$$TS_{offset} = (TS_{AGV} - TS_{Transfer}) - TS_{NAV}$$



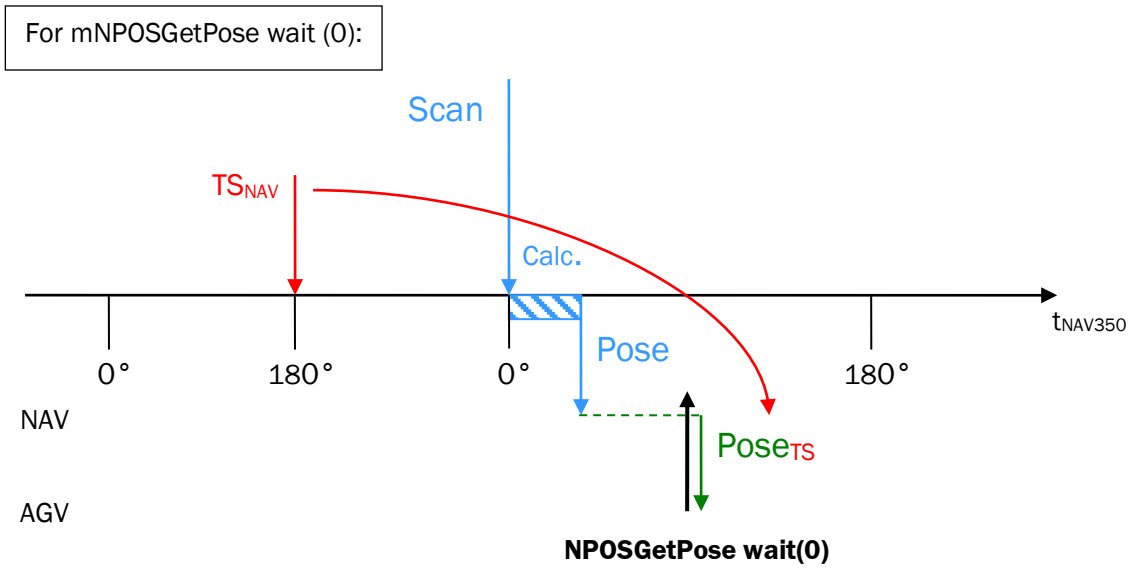
TS_{offset}	Resulting timing offset
TS_{AGV}	Timestamp of AGV
TS_{NAV}	Timestamp of NAV350
$TS_{Transfer}$	Transfer time via RS232 or Ethernet interface

1.8.2 Interpretation Timestamp/Position

The command mNPOSGetPose wait (1) requests the position of the NAV350 with the following timestamp (TS) at the 180° angle position. After completing the scan, the NAV350 emits the position information that was taken in the previous timestamp ($POSE_{TS}$).

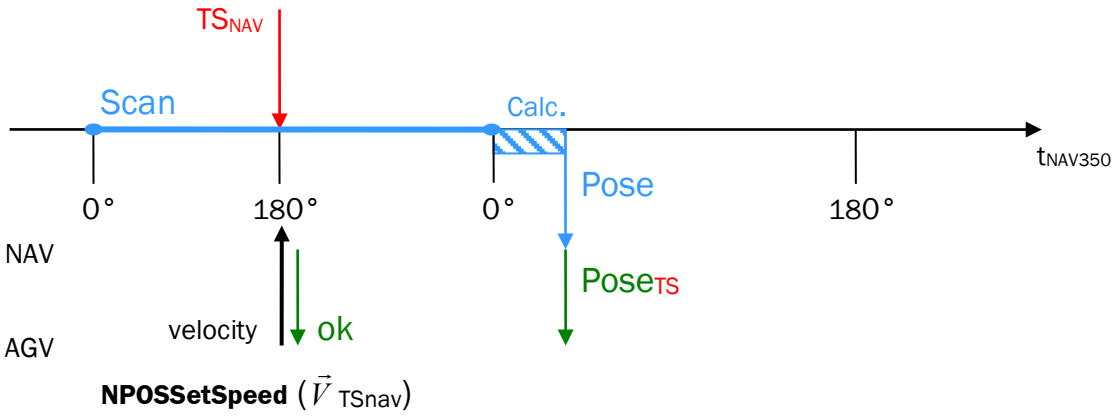


The command mNPOSGetPose wait (0) requests the last available position information of the NAV350 and gets the respond immediately with the timestamp (TS) from the previous 180° angle position.



1.8.3 Interpretation Timestamp / Speed with Synchronized AGV Timer (recommended)

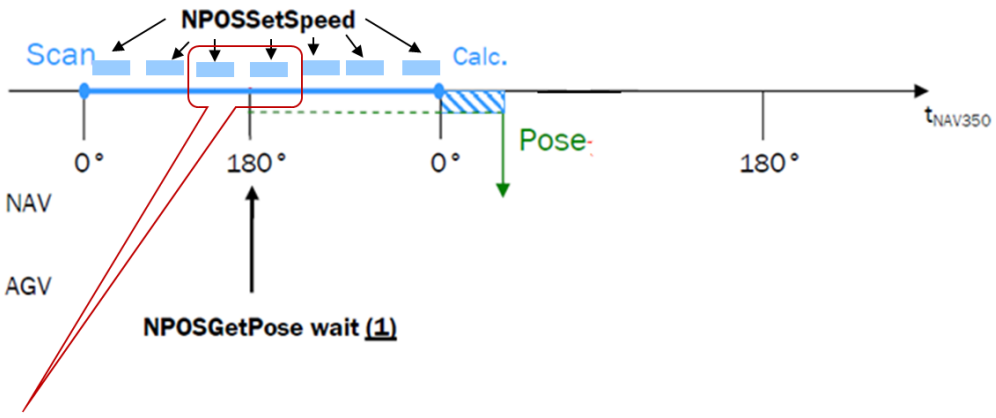
This formula shows how to calculate the value of TS_{NAV} based on the AGV timer.



$TS_{NAV} = TS_{VAGV} - TS_{Offset}$

1.8.4 Providing NPOSSetSpeed Based with a high frequency (40 Hz)

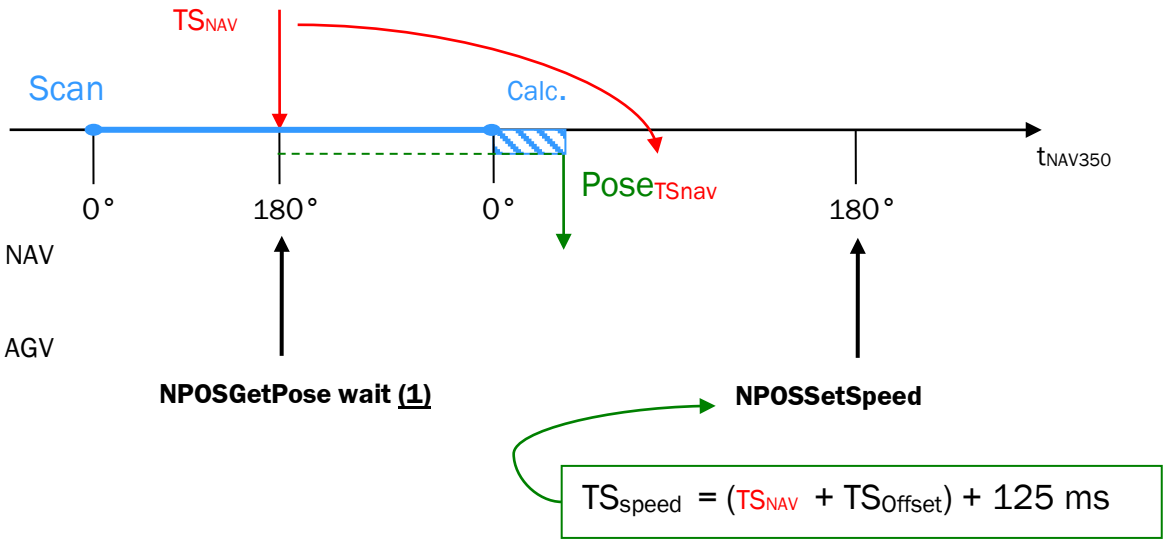
A simple way to provide the speed information is to send the information with the correct time stamp every 0,025sec (40Hz). The NAV350 will select the best 2 data which are close to the 180° point of time.



Used speed information

1.8.5 Timing NPOSSetSpeed Based on Constant Offset

The best point of time to provide the speed information of the AGV to the NAV350 is at the 180° position of the scanner head. The synchronized AGV timer calculates when it is time for the next speed information.



1.9 Error Messages (sFA)

All commands are initially interpreted by the CoLa-A or CoLa-B command interpreter, before the actual execution of the command follows. If the command interpreter determines an error, it responds with an error message in the form of "FA <Error number>". This is valid for all commands with direct variable access (“RA”/“WA"), events, methods and other commands.Common error numbers are described in the following table.

Error Number (ASCII)	Error Description
1	Access authorization for execution of the method is missing (low user level)
2	Unknown method name / index
3	Unknown variable / index
4	Variable or parameter is out of Range/Value
5	Invalid data
6	Unknown error
7	Memory Overflow
8	Parameter is missing, memory underflow
9	Unknown error type
A	Access authorization to write the variable is missing (low user level)
B	Unknown command for the name server
C	Unknown CoLa command
D	Synchronous method is still running (only one currently possible)
E	Flex Array is too large
F	Unknown Event name / index
10	CoLa-A Type overflow (value is higher than allowed for the type)
11	Illegal CoLa-A character (eg - or letter)
12	No message from the operating system SOPAS
13	No response from the operating system SOPAS
14	Internal error

Tab.1-8: Error messages of the command interpreter

2 Commands

2.1 Commands Accessible from all Modes

2.1.1 Command: Read Device Identification

Request Read

Command Syntax: **sRN DeviceIdent** (according: ST)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Read Device identification	String	**DeviceIdent**

Response

Command Syntax: **sRA DeviceIdent** *sizeName name sizeVersion version*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Read Device identification	String	**DeviceIdent**
SizeName	Length of name string	UInt_8	**0 ... 255**
Name	Device name	String	**...**
SizeVersion	Length of version string	UInt_8	**0 ... 255**
Version	SOPAS-Device version	String	**...**

2.1.2 Command: Read Serial Number

Request Read

Command Syntax: **sRN SerialNumber** (according: SS)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Read serial number	String	**SerialNumber**

Response

Command Syntax: **sRA SerialNumber** *sizeNumber serialNumber*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Read Serial number	String	**SerialNumber**
SizeNumber	Length serial number string	UInt_8	**0 ... 16**
SerialNumber	Serial number	String	**...**

2.1.3 Command: Read Measurement Firmware Version

Request Read

Command Syntax: **sRN MMDeviceInfo** (according: ST)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Read Measurement firmware Version	String	**MMDeviceInfo**

Response

Command Syntax: **sRA MMDeviceInfo** *version*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Read Measurement firmware Version	String	**MMDeviceInfo**
Version	Measurement firmware Version	String	**...**

2.1.4 Command: Read Application Version

Request Read

Command Syntax: **sRN FirmwareVersion**

(according: SV)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Read Application software version	String	**FirmwareVersion**

Response

Command Syntax: **sRA FirmwareVersion** *sizeVersion version*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Application software version	String	**FirmwareVersion**
SizeVersion	Length of Application version string	UInt_8	**0 ... 255**
Version	Application software version	String	**...**

2.1.5 Command: Set Current Layer

Set the current Layer for Positioning and Mapping.

Request Write

Command Syntax: **sWN NEVACurrLayer** *currLayer* (according: PL, PM)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS write by name)	String	**sWN**
Command	Set current Layer	String	**NEVACurrLayer**
CurrLayer	Layer number	UInt_16	**0 ... 319** **(Default: 0)**

Response

Command Syntax: **sWA NEVACurrLayer**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	**sWA**
Command	Set current Layer	String	**NEVACurrLayer**

Request Read

Command Syntax: **sRN NEVACurrLayer**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Read current Layer	String	**NEVACurrLayer**

Response

Command Syntax: **sRA NEVACurrLayer** *currLayer*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Read current Layer	String	**NEVACurrLayer**
CurrLayer	Current Layer	UInt_16	**0 ... 319**

2.1.6 Command: Define Reflector Identification Window

The identification window is used for identifying the reflectors in the continuous positioning mode.

In continuous positioning the NAV350 is in expectation of the approximate reflector positions.

The NAV350 sets an identification window with a radius of 300 mm (default radius) around each of the reflector coordinates in the current layer. If the NAV350 recognizes a reflector in this identification window it will allocate the measurement to this reflector and will use it the next time a position is determined.

The radius of the identification window can be changed with the command ***sWN NCORIdentWindow***. This enables the NAV350 to be optimized for extremely dynamic changes in velocity of the AGV and for challenging conditions resulting from faulty reflections.

The radius can be adjusted depending on the distance between the NAV350 and the reflector. The start and end points of an even function are transferred to the NAV350 to this purpose. The NAV350 uses this to calculate the respective radius of the identification window. The start and end point of the straight lines are defined at the distance *distLow* by the *winLow* radius and at the distance *distHigh* by the *winHigh* radius.

The radius of these two points can be set in the range of 100 ... 2000 mm.

Request Write

Command Syntax: **sWN NCORIdentWindow** *winLow winHigh distLow distHigh* (according: PF)

Command part	**Description**	**Type**	**Range/Value**
Command type	Request (SOPAS write by name)	String	**sWN**
Command	Define Reflector Identification Window	String	**NCORIdentWindow**
WinLow	Window radius in mm at *distLow*	UInt_16	**100 ... 2000 mm (Default: 300 mm)**
WinHigh	Window radius in mm at *distHigh*	UInt_16	**100 ... 2000 mm (Default: 300 mm)**
DistLow	Distance in mm at *winLow*	UInt_32	**500 ... 70 000 mm (Default: 500 mm)**
DistHigh	Distance in mm at *winHigh*	UInt_32	**500 ... 70 000 mm (Default: 70 000 mm)**

Response

Command Syntax: **sWA NCORIdentWindow**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	**sWA**
Command	Define Reflector Identification Window	String	**NCORIdentWindow**

Request Read

Command Syntax: **sRN NCORIdentWindow**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Read Reflector Identification Window	String	**NCORIdentWindow**

Response

Command Syntax: **sRA NCORIdentWindow** *winLow winHigh distLow distHigh*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Read Reflector Identification Window	String	**NCORIdentWindow**
WinLow	Window radius in mm at *distLow*	UInt_16	**100 ... 2000 mm**
WinHigh	Window radius in mm at *distHigh*	UInt_16	**100 ... 2000 mm**
DistLow	Distance in mm at *winLow*	UInt_32	**500 ... 70 000 mm**
DistHigh	Distance in mm at *winHigh*	UInt_32	**500 ... 70 000 mm**

2.1.7 Command: Configure Mapping

The command “Configure Mapping” sets the parameter to control the Mapping mode (refer command *Do Mapping* (p.65))

In the "Mapping" mode the NAV350 measures the reflector positions visible within its range in absolute coordinates. The measuring is related to one layer at a time, so the NAV350 must be informed of the layer (refer to the command *Set current Layer (p.20)*), about its own position and orientation in the absolute coordinate system (command *Configure Mapping*) and the size of the reflectors (refer to Command *Set Reflector Size (p.33)).*

If the parameter *negative* is set to 0, the NAV350 returns all measured reflectors. If the parameter *negative* is set to 1, the NAV350 returns **new** reflectors measured only in the current layer. The reflector data refers to the absolute coordinate system.

Request Write

Command Syntax: **sWN NMAPMapCfg** *mean negative x y phi*

Command part	**Description**	**Type**	**Range/Value**
Command type	Request (SOPAS write by name)	String	**sWN**
Command	Configure Mapping	String	**NMAPMapCfg**
Mean	Number of scans for averaging	UInt_8	**1...127** **(Default: 50)**
Negative	If the parameter is set to 0, the NAV350 responds all measured reflectors, If the parameter is set to 1, the NAV350 responds new reflectors. Existing reflectors in the current layer are ignored	Bool_1	**0 = positive-Mapping (Default)** **1 = negative-Mapping**
X	X-Position of the NAV350	Int_32	**-10 000 000...** **+10 000 000 mm** **(Default: 0)**
Y	Y-Position of the NAV350	Int_32	**-10 000 000...** **+10 000 000 mm** **(Default: 0)**
Phi	Heading of the NAV350	Int_32	**-360 000...** **+360 000 mdeg** **(Default: 0)**

Response

Command Syntax: **sWA NMAPMapCfg**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	**sWA**
Command	Configure Mapping	String	**NMAPMapCfg**

Request Read

Command Syntax: **sRN NMAPMapCfg**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Read Mapping configuration	String	**NMAPMapCfg**

Response

Command Syntax: **sRA NMAPMapCfg** *mean negative x y phi*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Read Mapping configuration	String	**NMAPMapCfg**
Mean	Number of scans for averaging	UInt_8	**1...127**
Negative	If the parameter is set to 0, the NAV350 responds all measured reflectors, If the parameter is set to 1, the NAV350 responds new reflectors. Existing reflectors in the current layer are ignored	Bool_1	**0 positive-Mapping** **1 negative-Mapping**
X	X-Position of the NAV350	Int_32	**-10 000 000...** **+10 000 000 mm**
Y	Y-Position of the NAV350	Int_32	**-10 000 000...** **+10 000 000 mm**
Phi	Orientation of the NAV350	Int_32	**-360 000...** **+360 000 mdeg**

2.1.8 Command: Set the Sliding Mean

Set the sliding mean for the positioning mode.

Request Write

Command Syntax: **sWN NPOSSlidingMean** *slidingMean* (according PN)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS write by name)	String	**sWN**
Command	Set sliding mean	String	**NPOSSlidingMean**
SlidingMean	Smoothing factor	UInt_8	**1 ... 63** **(Default: 1)**

Response

Command Syntax: **sWA NPOSSlidingMean**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	**sWA**
Command	Set sliding mean	String	**NPOSSlidingMean**

Request Read

Command Syntax: **sRN NPOSSlidingMean**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS write by name)	String	**sRN**
Command	Read sliding mean	String	**NPOSSlidingMean**

Response

Command Syntax: **sRA NPOSSlidingMean** *slidingMean*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	**sRA**
Command	Read sliding mean	String	**NPOSSlidingMean**
SlidingMean	Smoothing factor	UInt_8	**1 ... 63**

2.1.9 Command: Set Positioning Data Format

The command Set Positioning Data Format parameterizes the responds of the commands *NPOSGetData* (p.76) and *NPOSGetPose (p.72)*.

If *outputMode* is set to 1 the NAV350 extrapolates the calculated position data to the point in time of transmission. If set to 0 the NAV350 respond the position result instantly, valid at the timestamp in the pose responds (refer to the command **sMA mNPOSGetPose (p.76))**

Request Write

Command Syntax: **sWN NPOSPoseDataFormat** *outputMode showOptParam*

Command part	Description	Type	Range/Value
Command type	Request (SOPAS write by name)	String	**sWN**
Command	Set positioning data format	String	**NPOSPoseDataFormat**
OutputMode	If set to 1 the NAV350 extrapolates the calculated position data to the point in time of transmission.	Enum_8	**0 = normal** **1 = extrapolated (Default)**
ShowOptParam	Select the transmission of the optional parameter	Bool_1	**0 = suppressed (Default)** **1 = enabled**

Response

Command Syntax: **sWA NPOSPoseDataFormat**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	**sWA**
Command	Set positioning data format	String	**NPOSPoseDataFormat**

Request Read

Command Syntax: **sRN NPOSPoseDataFormat**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Read positioning Data Format	String	**NPOSPoseDataFormat**

Response

Command Syntax: **sRA NPOSPoseDataFormat** *outputMode showOptParam*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Read positioning Data Format	String	**NPOSPoseDataFormat**
OutputMode	If set to 1 the NAV350 extrapolates the calculated position data to the point in time of transmission.	Enum_8	**0 normal** **1 extrapolated**
ShowOptParam	Select the transmission of the optional parameter	Bool_1	**0 suppressed** **1 enabled**

2.1.10 Command: Set Landmark Data Format

The command *Set Landmark Data Format* controls the output format of landmarks, which can be retrieved with the method *NLMDGetData* (p.68) and method *mNPOSGetData* (p.76). After each reboot of the device all settings are back to the default settings.

Request Write

Command Syntax: **sWN NLMDLandmarkDataFormat** *format showOptParam landmarkFilter*

Command part	Description	Type	Range/Value
Command type	Request (SOPAS write by name)	String	**sWN**
Command	Set landmark data format	String	**NLMDLandmarkDataFormat**
Format	Select Cartesian or polar	Enum_8	**0 = Cartesian (Default)** **1 = polar**
ShowOptParam	Select transmission of the optional landmark parameters localID (local ID), ID (global ID), type, subtype , quality (reliability), timestamp, size, hitCount (number of hits) and meanEcho	Bool_1	**0 = suppress (Default)** **1 = enable**
LandmarkFilter	Select transmission of the optional landmark data in positioning mode with *NPOSGetData*	Enum_8	**0 = used** **1 = detected (Default)** **2 = expected landmark**

Response

Command Syntax: **sWA NLMDLandmarkDataFormat**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	**sWA**
Command	Selected landmark data format	String	**NLMDLandmarkDataFormat**

Request Read

Command Syntax: **sRN NLMDLandmarkDataFormat**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Read landmark data format	String	**NLMDLandmarkDataFormat**

Response

Command Syntax: **sRA NLMDLandmarkDataFormat** *format showOptParam landmarkFilter*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Read landmark data format	String	**NLMDLandmarkDataFormat**
Format	Response Cartesian or polar	Enum_8	**0 = Cartesian** **1 = polar**
ShowOptParam	Select transmission of the optional landmark parameters localID (local ID)ID (global ID), type, subtype , quality (reliability), timestamp, size, hitCount (number of hits) and meanEcho	Bool_1	**0 = suppress** **1 = enable**
LandmarkFilter	Select transmission of the optional landmark data in positioning mode with *NPOSGetData*	Enum_8	**0 = used** **1 = detected** **2 = expected landmark**

2.1.11 Command: Set Scan Data Format

The command *Set Scan Data Format* is used to control the output format of scan data (contour data) that uses the method *NLMDGetData* (p.68) and method *mNPOSGetData* (p.76). Default setting transmits all scan data. After each reboot of the devices all settings are reset to the default settings

Request Write

Command Syntax: **sWN NAVScanDataFormat** *dataMode showRSSI*

Command part	Description	Type	Range/Value
Command type	Request (SOPAS write by name)	String	**sWN**
Command	Set scan data format	String	**NAVScanDataFormat**
DataMode	Select distance and/or angle	Enum_8	**0 = none** **1 = distance only (Default)** **2 = distance and angle**
ShowRSSI	Set transmission of remission data (echo)	Bool_1	**0 = suppress (Default)** **1 = enable**

Response

Command Syntax: **sWA NAVScanDataFormat**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	**sWA**
Command	Settings of scan data format	String	**NAVScanDataFormat**

Request Read

Command Syntax: **sRN NAVScanDataFormat**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Read scan data format	String	**NAVScanDataFormat**

Response

Command Syntax: **sRA NAVScanDataFormat** *dataMode showRSSI*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Select scan data format	String	**NAVScanDataFormat**
DataMode	Select distance and/or angle	Enum_8	**0 = none** **1 = distance only** **2 = distance and angle**
ShowRSSI	Selected transmission of remission data (echo)	Bool_1	**0 = suppress** **1 = enable**

2.1.12 Command: Set Hardware Time Sync

This command parameterizes the hardware time sync output.

There are 2 modes to configure the hardware time synchronization:

1) cyclic

Cycle time in ms modulo base 2 is based on the global device timer. The hardware output toggles after the cycle determined with mask.

Mask = 10 ➔ The output toggles every time the global timer has a roll-over in the lower 10 bits.

Examples:

Mask	Cycle time
8	256 ms
10	1024 ms
15	32768 ms

The synchronization can be done, if the same mask is used on the data timestamps.

2) on request

If the command “Synchronize Timestamp” (p.47) is used, the output toggles when the NAV350 timestamp is loaded to the response of *mNAVGetTimestamp.*

Request Write

Command Syntax: **sWN NAVHardwareTimeSync** *mode mask*

Command part	Description	Type	Range/Value
Command type	Request (SOPAS write by name)	String	**sWN**
Command	Set Hardware Time Sync	String	**NAVHardwareTimeSync**
Mode	Setting cycling pulse or on command synchronize timestamp	Enum_8	**0 = cyclic pulse** **1 = on request (Default)**
Mask	Cycle time in bit (modulo base 2)	UInt_8	**10...20 bit** **(Default: 15 bit)**

Response

Command Syntax: **sWA NAVHardwareTimeSync**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	**sWA**
Command	Set Hardware Time Sync	String	**NAVHardwareTimeSync**

Request Read

Command Syntax: **sRN NAVHardwareTimeSync**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS write by name)	String	**sRN**
Command	Set Hardware Time Sync	String	**NAVHardwareTimeSync**

Response

Command Syntax: **sRA NAVHardwareTimeSync** *mode mask*

Command part	Description	Type	Range/Value
Command type	Request (SOPAS write by name)	String	**sRA**
Command	Set Hardware Time Sync	String	**NAVHardwareTimeSync**
Mode	Set cycling pulse or on command synchro-nize timestamp	Enum_8	**0 = cyclic pulse** **1 = on request**
Mask	Cycle time in bit (modulo base 2)	UInt_8	**10...20 bit**

2.1.13 Command: Set Reflector Size

This command parameterizes the reflector size.

Request Write

Command syntax: **sWN NLMDReflSize** *size*

Command part	Description	Type	Range/Value
Command type	Request (SOPAS write by name)	String	**sWN**
Command	Set reflector size	String	**NLMDReflSize**
Size	Size of reflector (diameter in case of cylindrical reflectors)	UInt_16	**1 ... 150 mm (Default: 80 mm)**

Response

Command syntax: **sWA NLMDReflSize**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	**sWA**
Command	Set reflector size	String	**NLMDReflSize**

Request Read

Command syntax: **sRN NLMDReflSize**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Read reflector size	String	**NLMDReflSize**

Response

Command syntax: **sRA NLMDReflSize** *size*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Read reflector size	String	**NLMDReflSize**
Size	Size of reflector (diameter in case of cylindrical reflectors)	UInt_16	**1 ... 150 mm**

2.1.14 Command: Set Reflector Type

This command parameterizes the reflector type (flat or cylindrical).

Request Write

Command syntax: **sWN NLMDReflType** *type*

Command part	Description	Type	Range/Value
Command type	Request (SOPAS write by name)	String	**sWN**
Command	Set reflector type	String	**NLMDReflType**
Type	Type of reflector	Enum_8	**1 = flat** **2 = cylindric (Default)**

Response

Command syntax: **sWA NLMDReflType**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	**sWA**
Command	Set reflector type	String	**NLMDReflType**

Request Read

Command syntax: **sRN NLMDReflType**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Get reflector type	String	**NLMDReflType**

Response

Command syntax: **sRA NLMDReflType** *type*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Get reflector type	String	**NLMDReflType**
Type	Type of reflector	Enum_8	**1 = flat** **2 = cylindric**

2.1.15 Command: Set Landmark Matching

In this command you can set the rules for detecting reflectors. Detection can be can be optimized in order to achieve a higher accuracy or all visible landmarks are accepted for a higher availability of reflectors.

Request Write

Command syntax: **sWN NLMDLandmarkMatching** *filter*

Command part	Description	Type	Range/Value
Command type	Request (SOPAS write by name)	String	**sWN**
Command	Set reflector type	String	**NLMDLandmarkMatching**
Filter	Choose between the acceptance of only optimal detected reflectors (0), optimal and from a flat angle detectedreflectors (1) or the two first options plus partly covered reflectors (2). The givcen specifications are only valid for the “optimal” setting	Enum_8	**0 = optimal (Default)** **1 = optimal +angle** **2 = optimal + angle + partly covered**

Response

Command syntax: **sWA NLMDLandmarkMatching**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	**sWA**
Command	Set reflector type	String	**NLMDLandmarkMatching**

Request Read

Command syntax: **sRN NLMDLandmarkMatching**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Get reflector type	String	**NLMDLandmarkMatching**

Response

Command syntax: **sRA NLMDLandmarkMatching** *filter*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Get reflector type	String	**NLMDLandmarkMatching**
Filter	Choose between the acceptance of only optimal detected reflectors (0), optimal and from a flat angle detectedreflectors (1) or the two first options plus partly covered reflectors (2). The givcen specifications are only valid for the "optimal" setting	Enum_8	**0 = optimal (Default)** **1 = optimal +angle** **2 = optimal + angle + partly covered**

2.1.16 Command: Set Sector Muting

By activating up to four sectors, it is possible to mute different angle ranges of the scanner.

Request Write

Command syntax: **sWN NLMDMutedSectors** *sector0 sector1 sector2 sector3*

Command part		Description	Type	Range/Value
Command type		Request (SOPAS read by name)	String	**sWN**
Command		Configuration of sector muting	String	**NLMDMutedSectors**
4 x sector	AngleFrom	Start angle of the sector	UInt_32	**0 ... 359 999 mdeg** **(Default: 0)**
	AngleTo	End angle of the sector	UInt_32	**0 ... 359 999 mdeg** **(Default: 0)**
	Active	Activation of the sector	Bool_1	**0 = inactive (Default)** **1 = active**

Response

Command syntax: **sWA NLMDMutedSectors**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sWA**
Command	Configuration of sector muting	String	**NLMDMutedSectors**

Request Read

Command syntax: **sRN NLMDMutedSectors**

Command part	**Description**	**Type**	**Range/Value**
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Configuration of sector muting	String	**NLMDMutedSectors**

Response

Command syntax: **sRA NLMDMutedSectors** *sector0 sector1 sector2 sector3*

Command part		**Description**	**Type**	**Range/Value**
Command type		Response (SOPAS read by name)	String	**sRA**
Command		Configuration of sector muting	String	**NLMDMutedSectors**
4 x sector	AngleFrom	Start angle of the sector	UInt_32	**0 ... 359 999 mdeg**
	AngleTo	End angle of the sector	UInt_32	**0 ... 359 999 mdeg**
	Active	Activation of the sector	Bool_1	**0 = inactive** **1 = active**

2.1.17 Command: Set Orientation Coordinate System

For devices mounted upside down the coordinate system may be mirrored. This variable serves to define the orientation. Default value for an upright mounted NAV350 is 1 = counterclockwise.

Request Write

Command syntax: **sWN NEVACoordOrientation** *direction* (according SU)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sWN**
Command	Set rotate direction relative to global coordinate system	String	**NEVACoordOrientation**
Direction	Set orientation of the coordinate system	Enum_8	**0 = clockwise** **1 = counterclockwise (Default)**

Response

Command syntax: **sWA NEVACoordOrientation**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sWA**
Command	Set orientation of the coordinate system	String	**NEVACoordOrientation**

Request Read

Command syntax: **sRN NEVACoordOrientation**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Set orientation of the coordinate system	String	**NEVACoordOrientation**

Response

Command syntax: **sRA NEVACoordOrientation** *direction*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Set orientation of the coordinate system	String	**NEVACoordOrientation**
Direction	Set orientation of the coordinate system	Enum_8	**0 = clockwise** **1 = counterclockwise**

2.1.18 Command: Set N Closest Reflectors

This command sets the maximum quantity of closest reflectors, which are used by the positioning. This has to be at least 3 reflectors. If N = 0, all valid reflectors are chosen. The quantity of “N closest reflectors” may be changed while landmark detection or positioning.

Request Write

Command syntax: **sWN NLMDnClosest** *nClosest* (according: PC)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sWN**
Command	Set N closest reflectors	String	**NLMDnClosest**
NClosest	Quantity of N closest reflectors	UInt_8	**0 ... 40** **(Default: 0)**

Response

Command syntax: **sWA NLMDnClosest**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sWA**
Command	Set N closest reflectors	String	**NLMDnClosest**

Request Read

Command syntax: **sRN NLMDnClosest**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Readout quantity N closest reflectors	String	**NLMDnClosest**

Response

Command syntax: **sRA NLMDnClosest** *nClosest*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Return quantity of N closest reflectors	String	**NLMDnClosest**
NClosest	Quantity of N closest reflectors	UInt_8	**0 ... 40**

2.1.19 Command: Set Action Radius

The action radius defines an area in the surrounding of NAV350. The reflectors within the limited area are preferred in positioning.

The parameters *rFr* and *rTo* define this area as a circular ring. With this command the action radius area may be changed while landmark detection or positioning.

Default: *rFr* = 500 mm, *rTo* = 70000 mm.

Request Write

Command syntax: **sWN NLMDActionRadius** *rFr rTo* (according: PO)

Command part	**Description**	**Type**	**Range/Value**
Command type	Request (SOPAS read by name)	String	**sWN**
Command	Set Action Radius	String	**NLMDActionRadius**
RFr	Minimum Actions Radius	UInt_32	**400 ... 70 000 mm** **(Default: 500)**
RTo	Maximum Action Radius	UInt_32	**400 ... 70 100 mm** **(Default: 70 000)**

Response

Command syntax: **sWA NLMDActionRadius**

Command part	**Description**	**Type**	**Range/Value**
Command type	Response (SOPAS read by name)	String	**sWA**
Command	Set action radius	String	**NLMDActionRadius**

Request Read

Command syntax: **sRN NLMDActionRadius**

Command part	**Description**	**Type**	**Range/Value**
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Request action radius	String	**NLMDActionRadius**

Response

Command syntax: **sRA NLMDActionRadius** *rFr rTo*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Request action radius	String	**NLMDActionRadius**
RFr	Minimum Actions Radius	UInt_32	**400 ... 70 000 mm**
RTo	Maximum Action Radius	UInt_32	**400 ... 70 100 mm**

2.1.20 Command: Set Reflector Threshold

To suppress false reflections, the NAV350 uses an internal, distance related threshold curve. It is based on device specific calibration on white and reflector foil (Diamond Grade 983-10).

By setting the threshold parameter it is possible to adapt the filter to specific environments.

A lowering of the threshold curve will also reduce the signal-to-noise ratio between reflectors and natural materials. Raising the threshold curve will reduce availability for reflector measurements.

0% means white remission, 100% reflector remission.

Request Write

Command syntax: **sWN NLMDReflThreshold** *percent* (according: RP, SP)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sWN**
Command	Set reflector threshold	String	**NLMDReflThreshold**
Percent	Reflector threshold value in percent	UInt_8	**0...100 in %** **(Default: 35)**

Response

Command syntax: **sWA NLMDReflThreshold**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sWA**
Command	Set reflector threshold	String	**NLMDReflThreshold**

Request Read

Command syntax: **sRN NLMDReflThreshold**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	**sRN**
Command	Request Reflector Threshold	String	**NLMDReflThreshold**

Response

Command syntax: **sRA NLMDReflThreshold** *percent*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	**sRA**
Command	Request Reflector Threshold	String	**NLMDReflThreshold**
Percent	Reflector threshold value in percent	UInt_8	**0...100 in %**

2.2 Method Call

Calling a method by command, instructs the NAV350 to process data or measurements. Methods can be called including data to be processed. The method responds with a structure of one or more return values. Whether a method can be called depends on the user level.

The NAV350 processes two types of methods that are synchronous or asynchronous methods:

- Synchronous commands **sMN** block the NAV350 until the results of the method are returned by **sAN**.
- Asynchronous commands **sMN** do not keep the NAV350 busy. The first response is a method acknowledge by **sMA,** indicating that the method has been started. After execution the scanner returns the result by **sAN**. Some asynchronous methods can be called with an input parameter, instructing the method to respond either immediately with the last valid value or with an acknowledgement followed by a new calculated result after processing.

2.3 Methods Accessible from all Modes except POWERDOWN

General methods can be selected from any mode except POWERDOWN.

2.3.1 Command: Set Operating Mode

This asynchronous method selects the NAV350 operating mode. The command will be acknowledged immediately by mNEVAChangeState with "**sMA**" followed by the response "**sAN** mNEVAChangeState" as soon as the mode has been changed. The default mode after switch-on is STANDBY.

Attention: From the POWERDOWN mode it is only possible to switch to the STANDBY mode and also to return to POWERDOWN, the mode has to be changed to STANDBY first.

Request

Command Syntax: **sMN mNEVAChangeState** *mode* (according: SA, DA, UA, RA, MA, PA, PN)

Command part	**Description**	**Type**	**Range/Value**
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Set operating mode	String	**mNEVAChangeState**
Mode	Number of valid operating mode.	Enum_8	**0 = power down** **1 = standby (Default)** **2 = mapping** **3 = landmark detection** **4 = navigation**

Response (Method Acknowledge)

Command Syntax: **sMA mNEVAChangeState**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method acknowledge by name)	String	**sMA**
Command	Asynchronous Method Set Operating mode	String	**mNEVAChangeState**

Response after Execution

Command Syntax: **sAN mNEVAChangeState** *errorCode mode*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS answer)	String	**sAN**
Command	Set Operating mode	String	**mNEVAChangeState**
ErrorCode	Error number	Enum_8	**0 = no error** **1 = invalid change** **2 = method break** **3 = unknown operating mode** **5 = timeout** **6 = method break; another command is active** **7 = general error**
Mode	Operating mode after execution	Enum_8	**0 = power down** **1 = standby** **2 = mapping** **3 = landmark detection** **4 = navigation**

2.3.2 Command: Set User Level

Selects a user level and transfers its password to the NAV350. The password is encoded (Hash Value)

User level	Password	Hash-Value
Operator	main	**B21ACE26**
Authorized client	client	**F4724744**

Request

Command Syntax: **sMN SetAccessMode** *newMode password*

Command part	Description	Type	Range/Value
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Set user level	String	**SetAccessMode**
NewModel	User level code	Int_8	**2 = operator** **3 = authorized client**
Password	Password Hash Value	UInt_32	**00000000 ... FFFFFFF**

Response

Command Syntax: **sAN SetAccessMode** *success*

Command part	Description	Type	Range/Value
Command type	Request (SOPAS answer)	String	**sAN**
Command	Set user level	String	**SetAccessMode**
Success	Mode successfully set	Bool_1	**0 = no** **1 = yes**

2.3.3 Command: Store Data Permanent

This command stores all settings of the NAV350 permanent, to be recalled after power on.

Request

Command Syntax: **sMN mEEwriteall**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Store data permanent	String	**mEEwriteall**

Response

Command Syntax: **sAN mEEwriteall** *success*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS answer)	String	**sAN**
Command	Store data permanent	String	**mEEwriteall**
Success	Shows if the command was successful.	Bool_1	**0 = no** **1 = yes**

2.3.4 Command: Synchronize Timestamp

This command returns the internal timestamp of the NAV350 to synchronize it with the vehicle controller's clock. The NAV350 clock is a free running *UInt_32* counter with 1ms increment. While reading the timestamp, a pulse is generated on the digital output to compensate the transmission time, if necessary.

Request

Command Syntax: **sMN mNAVGetTimestamp**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Sync Timestamp	String	**mNAVGetTimestamp**

Response

Command Syntax: **sAN mNAVGetTimestamp** *errorCode timestamp*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS answer)	String	**sAN**
Command	Sync Timestamp	String	**mNAVGetTimestamp**
ErrorCode	Error number	Enum_8	**0 = no error** **1 = invalid operating mode** **7 = general error**
Timestamp	Internal NAV350 timestamp	UInt_32	**0...4 294 967 295 ms (0xffffffff)**

2.3.5 Command: Break Asynchronous Method

Stops all running asynchronous method calls (sMN). The stopped methods respond with the error code *2 Method break.*

Request

Command Syntax: **sMN mNAVBreak**

(according: XB)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Break Asynchronous Method	String	**mNAVBreak**

Response

Command Syntax: **sAN mNAVBreak** *errorCode count*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method by name)	String	**sAN**
Command	Break Asynchronous Method	String	**mNAVBreak**
ErrorCode	Error number.	Enum_8	**0 = no error** **7 = general error**
Count	Number of terminated Methods	UInt_8	**0 ... 255**

2.3.6 Command: Device Reset

Runs a device reset and recalls all permanent stored parameters or sets it to default, if not stored.

This command responds an acknowledgement prior terminating the interface connection for reset.

Request

Command Syntax: **sMN mNAVReset** (according: XR)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Device reset	String	**mNAVReset**

Response

Command Syntax: **sAN mNAVReset** *errorCode*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method by name)	String	**sAN**
Command	Device reset	String	**mNAVReset**
ErrorCode	Error number	Enum_8	**0 = no error** **1 = invalid mode** **7 = general error**

2.4 Methods in STANDBY Mode

2.4.1 Command: Serial Interface Configuration

The method configures the serial interface. After the response the settings are valid without reboot. The command responds with the new setting.

Request

Command Syntax: **sMN mChangeSerialCfg** *baudrate databits parity stopbits* (according: SB)

Command part	**Description**	**Type**	**Range/Value**
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Serial configuration	String	**mChangeSerialCfg**
Baudrate	Baudrate	Enum_8	**0 = 19200** **1 = 38400** **2 = 57600** **3 = 115200 (Default)**
Data bits	Number of data bits in each character	UInt_8	**7 = 7 Data bits** **8 = 8 Data bits (Default)**
Parity	Parity check	Enum_8	**0 = none** **1 = odd** **2 = even (Default)** **3 = mark** **4 = space**
Stopbits	Stopbits	UInt_8	**1 (Default)**

Response

Command Syntax: **sAN mChangeSerialCfg** *errorCode baudrate databits/parity stopbits*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method by name)	String	**sAN**
Command	Serial configuration	String	**mChangeSerialCfg**
ErrorCode	Error number	Enum_8	**0 = no error (Default)** **1 = invalid baudrate** **2 = invalid databits** **3 = invalid parity** **4 = invalid stopbits** **5 = multiple invalid parameter** **6 = invalid databit/parity-combination** **7 = general error**
Baudrate	Baudrate	Enum_8	**0 = 19200** **1 = 38400** **2 = 57600** **3 = 115200 (Default)**
Data bits	Number of data bits in each character	UInt_8	**7 = 7 data bits** **8 = 8 data bits (Default)**
Parity	Parity check	Enum_8	**0 = none** **1 = odd** **2 = even (Default)** **3 = mark** **4 = space**
Stopbits	Stopbits	UInt_8	**1 (Default)**

2.4.2 Command: Change IP Configuration

This command changes the IP configuration of the Ethernet interface. The settings made here will be ignored if DHCP is enabled by *mEnableDHCP (p.54).* To activate the settings, it is necessary to run a device reset (*sMN mNAVReset (p.49)*. The command responds with the new settings.

Request

Command Syntax: **sMN mChangeIPCfg** *IP mask gateway*

Command part	**Description**	**Type**	**Range/Value**
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Change IP configuration	String	**mChangeIPCfg**
IP	IP-Address	4x UInt_8	**0...255 (4x)**
Mask	Subnetmask	4x UInt_8	**0...255 (4x)**
Gateway	Gateway	4x UInt_8	**0...255 (4x)**

Response

Command Syntax: **sAN mChangeIPCfg** *errorCode IP mask gateway*

Command part	**Description**	**Type**	**Range/Value**
Command type	Response (SOPAS method by name)	String	**sAN**
Command	Change IP configuration	String	**mChangeIPCfg**
ErrorCode	Error number	Enum_8	**0 = no error (Default)** **1 = DHCP active (warning)** **7 = general error**
IP	IP-Address	4x UInt_8	**0...255 (4x)**
Mask	Subnetmask	4x UInt_8	**0...255 (4x)**
Gateway	Gateway	4x UInt_8	**0...255 (4x)**

2.4.3 Command: Change Ethernet Configuration

This method configures the Ethernet interface. After the response, a reset of the device with the command *sMN mNAVReset (p.49)* is needed to activate the settings. The device returns the new settings.

Request

Command Syntax: **sMN mChangeEthrCfg** *speedDuplex*

Command part	Description	Type	Range/Value
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Change IP configuration	String	**mChangeEthrCfg**
SpeedDuplex	Interface speed and duplex mode	Enum_8	**0 = auto (Default)** **1 = 10MB half** **2 = 10MB full** **3 = 100MB half** **4 = 100MB full**

Response

Command Syntax: **sAN mChangeEthrCfg** *errorCode speedDuplex*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method by name)	String	**sAN**
Command	Change IP configuration	String	**mChangeEthrCfg**
ErrorCode	Error number	Enum_8	**0 = no error (Default)** **1 = invalid value** **7 = general error**
SpeedDuplex	Interface speed and duplex mode	Enum_8	**0 = auto (Default)** **1 = 10MB half** **2 = 10MB full** **3 = 100MB half** **4 = 100MB full**

2.4.4 Command: DHCP Enabled / Disabled

This method enables or disables DHCP. If DHCP is enabled, the IP configuration is ignored. If the command does not change the DHCP state, the command responds "no error" and the existing state. If the command changes the DHCP state, the NAV350 responds the new settings and runs a reboot. After the reboot the NAV350 obtains a new IP Address by DHCP. If DHCP is enabled, but no DHCP server available, the static IP address is used.

Request

Command Syntax: **sMN mEnableDHCP** *DHCP*

Command part	Description	Type	Range/Value
Command type	Request (SOPAS method by name)	String	**sMN**
Command	DHCP enable / disable	String	**mEnableDHCP**
DHCP	Enable/Disable DHCP	Bool_1	**0 = disabled (Default)** **1 = enabled**

Response

Command Syntax: **sAN mEnableDHCP** *errorCode DHCP*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method by name)	String	**sAN**
Command	DHCP enable / disable	String	**mEnableDHCP**
ErrorCode	Error number	Enum_8	**0 = no error (Default)** **7 = general error**
DHCP	Enable/Disable DHCP	Bool_1	**0 = disabled (Default)** **1 = enabled**

2.4.5 Command: Select Serial Host Protocol

(Introduced since Firmware V1.22.1 for standard NAV350, NAV350S01 and S1.21.1 for NAV350S02)

For the communication via the serial host interface it is possible to change between the Cola-A format or to replace a NAV200 to select the binary NAV200 prototocol.

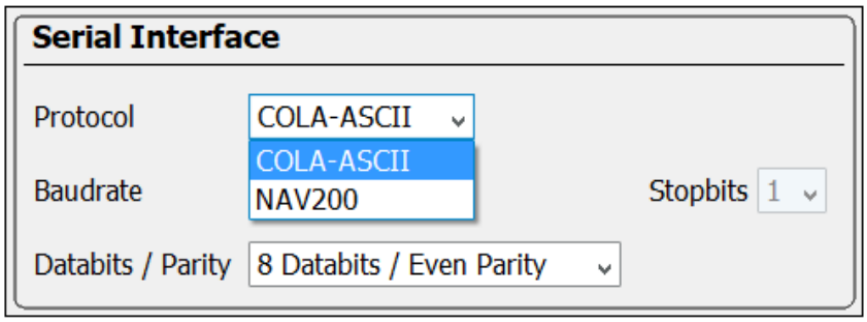
Default settings:

NAV350 : COLA-ASCII protocol

NAV350S01 : COLA-ASCII protocol

NAV350S02 : NAV200 protocol

By SOPAS ET the selection may be done as:



Request

Command Syntax: **sWN SIHstActProt** protocol

Command part	Description	Type	Range/Value
Command type	Write (SOPAS method by name)	String	**sWN**
Command	Set active protocol of serial host interface	String	**SIHstActProt**
Protocol	Protocol	Enum_8	**0 : Cola-A** **1 : NAV200-protocol**

Response

Command Syntax: **sWA SIHstActProt**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	**sWA**
Command	Set active protocol of serial host interface	String	**SIHstActProt**

The selection in both cases takes place without automatic saving.

The switch is done after Logout ("sMN Run").

2.4.6 Command Add Landmark

This command adds 1 to 50 reflector landmarks to the NAV350 layout memory.

The NAV350 responds with a list of assigned global reflector landmark IDs after execution.

Request :

Command Syntax: **sMN mNLAYAddLandmark** *landmarkData {x y type subtype size layerID {ID}}*

(according: DR, SI, RS, BS)

Command part			**Description**	**Type**	**Range/Value**
Command type			Request (SOPAS method by name)	String	**sMN**
Command			Add landmark	String	**mNLAYAddLandmark**
LandmarkData			Number of landmarks follow	UInt_16	**1...50**
Landmark data	X		X-Position	Int_32	**-10 000 000 ...** **+10 000 000 mm**
	Y		Y-Position	Int_32	**-10 000 000 ...** **+10 000 000 mm**
	Type		Landmark type	Enum_8	**0 = undefined** **1 = reflector (fix)**
	Subtype		Reflector type	Enum_8	**0 = undefined** **1 = flat** **2 = cylindric**
	Size		Size: Diameter or width	UInt_16	**0...200 mm** **(Default: 80 mm)**
	LayerID		Layer IDs assigned, 1..3 follow	UInt_16	**1...3**
	Layer	ID	Layers ID	UInt_16	**0...319**

Response

Command Syntax: **sAN mNLAYAddLandmark** *errorCode landmarkData {globalID}*

Command part		**Description**	**Type**	**Range/Value**
Command type		Response (SOPAS method by name)	String	**sAN**
Command		Add landmark	String	**mNLAYAddLandmark**
ErrorCode		Command accepted if errorCode = 0	Enum_8	**0 = no error (default)** **1 = invalid mode** **3 = invalid data** **7 = general error**
LandmarkData		Number of landmarks	UInt_16	**1...50**
LMD	GlobalID	Assigned global ID	UInt_16	**0...11 999**

2.4.7 Command: Edit Landmark

This Command edits 1 to 50 reflector landmarks. If the global ID is not assigned, a new landmark will be added in the layout memory of the NAV350. If the global ID is already assigned to a reflector landmark, the reflector of this global ID will be modified.

Request

Command Syntax: **sMN mNLAYSetLandmark** *landmarkData {globalID X Y type subtype size layerID {ID}}*
(according: SC, RS, BS)

Command part			**Description**	**Type**	**Range/Value**
Command type			Request (SOPAS method by name)	String	**sMN**
Command			Edit landmark	String	**mNLAYSetLandmark**
LandmarkData			Number of landmarks follow	UInt_16	**1...50**
Landmark data	GlobalID		Global ID of the landmark to add or to modify.	UInt_16	**0...11 999**
	X		X-Position	Int_32	**-10 000 000 ...** **+10 000 000 mm**
	Y		Y-Position	Int_32	**-10 000 000 ...** **+10 000 000 mm**
	Type		Landmark type	Enum_8	**0 = undefined** **1 = reflector**
	Subtype		Reflector type	Enum_8	**0 = undefined** **1 = flat** **2 = cylindric**
	Size		Size: Diameter or width	UInt_16	**0...200 mm** **(Default 80 mm)**
	LayerID		Layer IDs assigned, 1..3 follow	UInt_16	**1...3**
	Layer	ID	Layer ID	UInt_16	**0...319**

Response

Command Syntax: **sAN mNLAYSetLandmark** *errorCode*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method by name)	String	**sAN**
Command	Edit landmark	String	**mNLAYSetLandmark**
ErrorCode	Command accepted if errorCode = 0	Enum_8	**0 = no error (default)** **1 = invalid mode** **3 = invalid data** **7 = general error**

2.4.8 Command: Delete Landmark

This Command deletes 1 to 50 reflector landmarks.

Request

Command Syntax: **sMN mNLAYDelLandmark** *landmarkData {globalID}* (according: SD)

Command part		Description	Type	Range/Value
Command type		Request (SOPAS method by name)	String	**sMN**
Command		Delete landmark	String	**mNLAYDelLandmark**
LandmarkData		Number of landmarks follow for delete	UInt_16	**1...50**
LMD	GlobalID	Global ID of the landmark to be deleted	UInt_16	**0...11 999**

Response

Command Syntax: **sAN mNLAYDelLandmark** *errorCode*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method by name)	String	**sAN**
Command	Delete landmark	String	**mNLAYDelLandmark**
ErrorCode	Command accepted if errorCode = 0	Enum_8	**0 = no error** **1 = invalid mode** **3 = invalid Data** **7 = general error**

2.4.9 Command: Read Landmark

This Command reads 1 to 50 reflector landmarks of the layout memory of the NAV350.

Request

Command Syntax: **sMN mNLAYGetLandmark** *landmarkData {globalID}* (according: UR, SR, RG, BR)

Command part		**Description**	**Type**	**Range/Value**
Command type		Request (SOPAS method by name)	String	**sMN**
Command		Read landmark	String	**mNLAYGetLandmark**
LandmarkData		Set the number of landmarks for read	UInt_16	**1...50**
LMD	GlobalID	Global ID of the transmitted landmark	UInt_16	**0...11 999**

Response

Command Syntax: **sAN mNLAYGetLandmark** *errorCode landmarkData {globalID x y type subtype size layerID {ID}}*

Command part		**Description**	**Type**	**Range/Value**
Command type		Response (SOPAS method by name)	String	**sAN**
Command		Read landmark	String	**mNLAYGetLandmark**
ErrorCode		Command accepted if errorCode = 0.	Enum_8	**0 = no error** **1 = invalid mode** **3 = invalid Data** **7 = general error**
LandmarkData		Number of landmarks follow	UInt_16	**1...50**
Landmark Data	GlobalID	Global ID of the landmark for read.	UInt_16	**0...11 999**
	X	X-Position	Int_32	**-10 000 000 ...** **+10 000 000 mm**
	Y	Y-Position	Int_32	**-10 000 000 ...** **+10 000 000 mm**
	Type	Landmark type	Enum_8	**0 = undefined** **1 = reflector (fix)**
	Subtype	Reflector type	Enum_8	**0 = undefined** **1 = flat** **2 = cylindric**

	Size		Size: Diameter or width	UInt_16	**1...150 mm** **(Default: 80 mm)**
	LayerID		Layer IDs assigned, 1..3 follow	UInt_16	**1...3**
	Layer	ID	Layer ID	UInt_16	**0...319**

2.4.10 Command : Read Layer

This Command reads the global IDs of all reflector landmarks in the current layer.

Request

Command Syntax: **sMN mNLAYGetLayer** *layerID*

Command part	**Description**	**Type**	**Range/Value**
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Get Layer	String	**mNLAYGetLayer**
LayerID	ID of the Layer	UInt_16	**0...319**

Response

Command Syntax: **sAN mNLAYGetLayer** *errorCode globalID{globalID}*

Command part		**Description**	**Type**	**Range/Value**
Command type		Response (SOPAS method by name)	String	**sAN**
Command		Get Layer	String	**mNLAYGetLayer**
ErrorCode		Command accepted if errorCode = 0	Enum_8	**0 = no error (Default)** **1 = invalid mode** **3 = invalid Data** **7 = general error**
GlobalID		Number of global Reflector IDs follow	UInt_16	**0...12 000**
Data	GlobalID	Global ID of the landmark	UInt_16	**0...11 999**

2.4.11 Command: Read Layout

This Command reads all the global IDs of all landmarks in the layout memory of the NAV350

Request

Command Syntax: **sMN mNLAYGetLayout**

Command part	**Description**	**Type**	**Range/Value**
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Read Layout	String	**mNLAYGetLayout**

Response

Command Syntax: **sAN mNLAYGetLayout** *errorCode globalID {globalID}*

Command part		**Description**	**Type**	**Range/Value**
Command type		Response (SOPAS method by name)	String	**sAN**
Command		Read Layout	String	**mNLAYGetLayout**
ErrorCode		Command accepted if errorCode = 0	Enum_8	**0 = no error (Default)** **1 = invalid mode** **3 = invalid Data** **7 = general error**
GlobalID		Number of IDs follow	UInt_16	**0...12 000**
Data	GlobalID	Global ID of existing landmarks	UInt_16	**0...11 999**

2.4.12 Command: Erase Layout

This Command erases all landmarks in the layout memory of the NAV350

Request :

Command Syntax: **sMN mNLAYEraseLayout** *erase*

Command part	**Description**	**Type**	**Range/Value**
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Erase Layout	String	**mNLAYEraseLayout**
Erase	Select the Memory for erase	Enum_8	**0 = ram** **1 = ram+flash**

Response

Command Syntax: **sAN mNLAYEraseLayout** *errorCode*

Command part	**Description**	**Type**	**Range/Value**
Command type	Response (SOPAS method by name)	String	**sAN**
Command	Erase Layout	String	**mNLAYEraseLayout**
ErrorCode	Command accepted if errorCode = 0	Enum_8	**0 = no error** **1 = invalid mode** **3 = invalid data** **7 general error**

2.4.13 Command: Store Layout Permanent

This Command transfers all landmarks in the layout memory of the NAV350 to the flash memory for permanent storage.

Request

Command Syntax: **sMN mNLAYStoreLayout**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Store Layout permanent	String	**mNLAYStoreLayout**

Response

Command Syntax: **sAN mNLAYStoreLayout** *errorCode*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method by name)	String	**sAN**
Command	Store Layout permanent	String	**mNLAYStoreLayout**
ErrorCode	Command accepted if errorCode = 0	Enum_8	**0 = no error (Default)** **1 = invalid mode** **7 = general error**

2.5 Methods in MAPPING Mode

2.5.1 Command: Do Mapping

In the "Mapping" mode the NAV350 measures the reflector positions visible within its range in absolute coordinates. The measuring is related to one layer at a time, so the NAV350 must be informed of the layer (refer to the command *Set current Layer (p.20)*), about its own position and orientation in the absolute coordinate system (command *Configure Mapping*) and the size of the reflectors (refer to command *Set Reflector Size (p.33)).*

The Method *Do Mapping* acknowledges the start of the Method instantly and responds the results after execution.

Request

Command Syntax: **sMN mNMAPDoMapping** (according: MS, MM, MN, MR)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Do Mapping	String	**mNMAPDoMapping**

Response (Method Acknowledge)

Command Syntax: **sMA mNMAPDoMapping**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method acknowledge)	String	**sMA**
Command	Do Mapping	String	**mNMAPDoMapping**

Response when Mapping is Executed

Command Syntax: **sAN mNMAPDoMapping** *errorCode landmarkData[...]*

Command part				**Description**	**Type**	**Range/Value**
Command type				Response (SOPAS method by name)	String	**sAN**
Command				Do Mapping	String	**mNMAPDoMapping**
ErrorCode				Error code	Enum_8	**0 = no error (Default)** **1 = wrong operating mode** **2 = asynchrony Method terminated** **5 = timeout** **6 = method already active** **7 = general error**
LandmarkData				Landmark data available	UInt_16	**0 = no data** **1 = landmark data follow**
LMD	Landmark-Filter			Data of seen landmarks	Enum_8	**0 = used** **1 = seen landmark (fix)** **2 = expected**
	Reflectors			Number of reflector landmark follow	UInt_16	**0...40**
	Landmark Data	Cart		Cartesian coordinate data follow	UInt_16	**0 = no Cartesian data follow** **1 = Cartesian data follow**
		Cart	X	X-Coordinate	Int_32	**-70 000...** **+70 000 mm**
			Y	Y-Coordinate	Int_32	**-70 000...** **+70 000 mm**
		Polar		Polar coordinate data follow	UInt_16	**0 = no polar data follow (fix)** **1 = polar data follow**
		Polar	Dist	Distance	UInt_32	
			Phi	Orientation	Uint_32	
		OptReflectorData		Optional landmark follow	UInt_16	**1 = optional data follow (fix)**
		Optional Reflector	LocalID	Local ID	UInt_16	**0...40**
			GlobalID	Global ID	UInt_16	**0...11 999**

			Type	Landmark type	Enum_8	**0 = undefined** **1 = reflector**
			Sub-type	Reflector type	Enum_16	**0 = undefined** **1 = flat** **2 = cylindrical**
			Quality	Reserved	UInt_16	**0 = Reserved**
			Time-stamp	Timestamp	UInt_32	**0...4 294 967 295 ms** **(0xffffffff)**
			Size	Size	UInt_16	**1...150 mm** **(Default: 80mm)**
			HitCount	Number of hits on the landmark	UInt_16	**1...1440**
			Mean-Echo	Mean echo value	UInt_16	**0...1023**
			Index-Begin	Scan index of the landmark start point	UInt_16	**0...1439**
			Index-End	Scan index of the landmark end point	UInt_16	**0...1439**

2.6 Methods in LANDMARK Mode

2.6.1 Command: Get Landmark data

This command responds the currently recognized landmarks of the NAV350, including optional scan data. The configuration of the transferred landmark parameter is set with the command *NLMDLandmarkData-Format* (p.28). By the *wait* parameter, a choice between the synchronous request of the last seen landmarks and the asynchronous request of new landmarks can be made

Request

Command Syntax: **sMN mNLMDGetData** *wait mask* (according RD, RK)

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method acknowledge by name)	String	**sMN**
Command	Get detected landmarks	String	**mNLMDGetData**
Wait	Set last detected landmark (0) or new landmarks (1)	Bool_1	**0 = last detected landmark** **1 = wait for new landmarks**
Mask	Select data format	Enum_8	**0 = reflectors** **1 = reflectors +scan**

Response (Method Acknowledge, the asynchronous method has started)

Command Syntax: **sMA mNLMDGetData**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method acknowledge by name)	String	**sMA**
Command	Asynchronous request of new landmarks	String	**mNLMDGetData**

Response after Executing Landmark Data Request

Command syntax: **sAN mNLMDGetData** *version errorCode wait countOfLandmarkData [LandmarkData countOptLandmarkData [optLandmarkData]] countOfScanData [scanDaten]*

Command part				**Description**	**Type**	**Range/Value**
Command type				Response (SOPAS method by name)	String	**sAN**
Command				Respond landmarks	String	**mNLMDGetData**
Version				Version of this response	UInt_16	**0000 ... FFFF**
ErrorCode				Error number	Enum_8	**0 = no error** **1 = wrong operating mode** **2 = asynchronous method terminated** **3 = invalid data** **5 = timeout** **6 = method already active** **7 = general error**
Wait				Response last detected landmark (0) or new landmarks	Bool_1	**0 = synchronous: last landmarks detected** **1 = asynchronous: new landmarks**
Mask				Data format	Enum_8	**0 = reflectors** **1 = reflectors + scan**
LandmarkData				Landmark transmission	UInt_16	**0 = no landmark data** **1 = landmark data**
Landmark data	Landmark-Filter			Indicate landmarks	Enum_8	**0 = used** **1 = detected** **2 = expected**
	Reflectors			Number of reflector data	UInt_16	**0...40** **reflector data sets**
	Landmark Data	Cart		Indicates landmark data Cartesian or not. Selectable by NLMDRelfDataFormat	UInt_16	**0 = not Cartesian** **1 = Cartesian**
		Cart.	X	X-coordinate	Int_32	**-70 000...** **+70 000 mm**

			Y	Y-coordinate	Int_32	**-70 000...** **+70 000 mm**
		Polar		Indicates landmark data polar or not. Selectable NLMDRelfDataFormat	UInt_16	**0 = not polar** **1 = polar coordinates**
		Polar	Dist	Distance	UInt_32	**0...70 000 mm**
			Phi	Angle in mdeg degree	UInt_32	**0...360 000 mdeg**
		optLandmarkData		Indicates response of optional landmarks selectable by showOptParam in NLMDRelfDataFormat	UInt_16	**0 = no optional data** **1 = optional reflector data**
		Optional Reflector Data	LocalID	Local ID	UInt_16	**0...40**
			GlobalID	Global ID	UInt_16	**0...11 999**
			Type	Landmark type	Enum_8	**0 = undefined** **1 = reflector**
			Sub-type	Reflector type	Enum_16	**0 = undefined** **1 = flat** **2 = cylindric**
			Quality	Quality (reliability)	UInt_16	**reserved**
			Time-stamp	timestamp	UInt_32	**0...** **4 294 967 295ms (0xffffffff)**
			Size	Size	UInt_16	**1...150 mm (Default: 80mm)**
			HitCount	Number of hits	UInt_16	**1...1440**
			Mean-Echo	Mean Echo Amplitude	UInt_16	**0...1023**
			Index-Begin	Start Index of the reflector	UInt_16	**0...1439**
			Index-End	End Index of the reflector	UInt_16	**0...1439**

NAVscanData			Indicates the number of output channels of the Scan data; Channel with 32 bit values	UInt_16	**0 = no scan data** **1 = one output channel for scan data (distance)** **2 = two output channels for scan data (distance + echo)**
Scan Data	AContentType		Content of the channel.	String	**DIST1 : distance** **ANGL1 : angle**
	DScaleFactor		Multiplier	Real	**1 (for NAV350 always)**
	DScaleOffset		Offset	Real	**0 (for NAV350 always)**
	DiStartAngle		Start angle of the Scans	Int_32	**0...+360 000 mdeg**
	UiAngleRes		Angular step	UInt_16	**1/1000 degree**
	UiTimestampStart		Timestamp Start of scan	UInt_32	**ms**
	AData		Number of scan data	UInt_16	**0...1440**
	Scan	AData	Scan data 1 to n. DIST in mm, ANGL in 1/10.000 degree	UInt_32	**00000000... FFFFFFFF**
NAVRemissionData			Indicates a 16 bit channel is enabled for remission data	UInt_16	**0 = no remission data** **1 = one channel for remission data**
Remissions-Scan Data	AContentType		Type of data	String	**RSSI1 Remission**
	DScaleFactor		Multiplier	Real	**1 (for NAV350 always)**
	DScaleOffset		Offset	Real	**0 (for NAV350 always)**
	DiStartAngle		Start angle of the Scans	Int_32	**0...+360 000 mdeg**
	UiAngleRes		Angular step	UInt_16	**1/1000 degree ; 250 mdeg (fix)**
	UiTimestampStart		Timestamp Start of scan	UInt_32	**ms**
	AData		Number of scan data	UInt_16	**0...1440**
	Scan	AData	Scan data 1 to n	UInt_16	**0000...FFFF**

2.7 Methods in NAVIGATION Mode

2.7.1 Command: Position Request

In the “Navigation” mode (determining position) the NAV350 continuously calculates its own position and orientation from the known position of the reflectors in an absolute coordinate system.

This position data is made available for transmission. The NAV350 takes account of its own onward movement by means of consistently using the velocity vector, i.e. the system delivers its position and direction extrapolated at the point in time of data transmission or related to the timestamp provided in the response.

If beside the pose data also the contour (scan) data and/or relative landmark data is needed, the command *Position Data Request* (*mNPOSGetData* , p,76) has to be used.

The method *mNPOSGetPose* acknowledges the start of the Method instantly and responds the results after execution.

To configure the output data refer to the command *NPOSPoseDataFormat (p.27).*

Request

Command Syntax: **sMN mNPOSGetPose** *wait* (according: PP, PV, Pv, Pw)

Command part	**Description**	**Type**	**Range/Value**
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Position Request	String	**mNPOSGetPose**
Wait	If set to 1 the NAV350 waits for the next calculated position data. If set to 0 the NAV350 respond the position result instantly, valid at the timestamp in the pose response	Bool_1	**0 = instantly last pose result** **1 = wait for next pose result**

Response (Method Acknowledge, indicates *NPOSGetPose* has started)

Command Syntax: **sMA mNPOSGetPose**

Command part	**Description**	**Type**	**Range/Value**
Command Type	Response (SOPAS method acknowledge)	String	sMA
Command	Get pose	String	mNPOSGetPose

Response after Executing *mNPOSGetPose*

Command Syntax: **sAN mNPOSGetPose** *version errorCode wait poseData [pose data [x y phi] optPoseData [outputMode timestamp meanDev navMode infoState quantUsedReflectors]]*

Command part		**Description**	**Type**	**Range/Value**
Command type		Response (SOPAS method by name)	String	**sAN**
Command		GetPose	String	**mNPOSGetPose**
Version		Version of POSE Data	UInt_16	**0000 ... FFFF**
ErrorCode		Error code	Enum_8	**0 = no error** **1 = wrong operating mode** **2 = asynchrony Method terminated** **3 = invalid data** **4 = no position available** **5 = timeout** **6 = method already active** **7 = general error**
Wait		If set to 1 the NAV350 waits for the next calculated position data. If set to 0 the NAV350 respond the position result instantly, valid at the timestamp in the pose response	Bool_1	**0 = synchronous: last position result** **1 = asynchronous: new calculated position**
PoseData		Set to1, if pose data follow	UInt_16	**0 = no pose data** **1 = pose data follow**
Pose Data	X	X-Position	Int_32	**-10 000 000 ...** **+10 000 000 mm**
	Y	Y-Position	Int_32	**-10 000 000 ...** **+10 000 000 mm**
	Phi	Orientation	UInt_32	**0...360000 mgrad**

	OptPoseData		Set to 1, if optional pose data follow	UInt_16	**0 = no optional data** **1 = optional data follow**
	OPD	output-Mode	Data are instantly or extrapolated	Enum_8	**0 = instantly** **1 = extrapolated**
		Time-stamp	Timestamp	UInt_32	**0...4 294 967 295 ms** **(0xffffffff)**
		Mean-Dev	Mean deviation	Int_32	**0... 2000 mm**
		Nav-Mode	Position mode	Enum_8	**0 = initial positioning** **1 = continuous positioning** **2 = virtual positioning** **3 = positioning stopped** **4 = position invalid** **5 = external**
		InfoState	Diagnosis Information	UInt_32	**Information is shown in the table below**
		NumUsedReflectors	Number of used Reflectors	UInt_8	**0...40**

InfoState

Bit					Description
0x	00	00	01	00	No external speed available
0x	00	00	02	00	No internal speed available
0x	00	00	04	00	Internal prediction used
0x	00	00	08	00	Simple prediction used
0x	00	00	10	00	No prediction used
0x	00	00	20	00	No speed compensation done
0x	00	00	40	00	Not enough landmarks in layer
0x	00	00	80	00	No landmarks available
0x	00	01	00	00	Not enough landmarks in layer for initialization
0x	00	02	00	00	Not enough landmarks available
0x	00	04	00	00	Less than 3 landmarks used
0x	00	08	00	00	Positioning failed
0x	00	10	00	00	Positioning reinitialized with internal speed
0x	00	20	00	00	Positioning reinitialized
0x	00	40	00	00	Positioning reinitialized
0x	00	00	00	01	Position is extrapolated
0x	00	00	00	02	Filter operating radii was skipped
0x	00	00	00	04	Filter sector muting was skipped
0x	00	00	00	08	Filter NClosest was skipped
0x	01	00	00	00	Speed computation done
0x	02	00	00	00	Prediction done
0x	04	00	00	00	Correlation done
0x	08	00	00	00	Reinitialized
0x	10	00	00	00	Reinitialized
0x	20	00	00	00	Landmark selection done
0x	40	00	00	00	Positioning done
0x	80	00	00	00	Positioning finished

2.7.2 Command: Position Data Request

This command operates equal to the command position request *mNPOSGetPose (p.72)* with the extension to also provide the scan data and relative landmark data.

Request

Command Syntax: **sMN mNPOSGetData** *wait mask* (according: PP, PV, Pv, Pw)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Position data request	String	**mNPOSGetData**
Wait	If set to 1 the NAV350 waits for the next calculated position data. If set to 0 the NAV350 respond the position result instantly, valid at the timestamp in the pose response	Bool_1	**0 = instantly last pose result** **1 = wait for next pose result**
Mask	Select the set of data	Enum_8	**0 = pose+reflectors** **1 = pose+scan** **2 = pose+reflectors+ scan**

Response (Method Acknowledge, indicates *mNPOSGetData* has started)

Command Syntax: **sMA mNPOSGetData**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method acknowledge)	String	**sMA**
Command	Position data request	String	**mNPOSGetData**

Response after Executing Position Data Request

Command Syntax: **sAN mNPOSGetData** *version errorCode wait poseData [x y phi optPoseData [outputMode timestamp meanDev navMode infoState quantUsedReflectors]] landmarkData [landmarkFilter reflectors [cart [X Y] polar [D Phi] optLandmarkData [optLandmarkData...]]] scanData [contentType scaleFactor scaleOffset startAngle angleRes data [aData]]*

Command part		Description	Type	Range/Value
Command type		Response (SOPAS method by name)	String	**sAN**
Command		Position data request	String	**mNPOSGetData**
Version		Version of positioning data	UInt_16	**0000 ... FFFF**
ErrorCode		ErrorCode	Enum_8	**0 = no error** **1 = wrong operating mode** **2 = asynchrony Method terminated** **3 = invalid data** **4 = no position available** **5 = timeout** **6 = method already active** **7 = general error**
Wait		If set to 1 the NAV350 waits for the next calculated position data. If set to 0 the NAV350 respond the position result instantly, valid at the timestamp in the pose response	Bool_1	**0 = synchronous: last position result** **1 = asynchronous: new calculated position**
Mask		Selected set of data	Enum_8	**0 = pose+reflectors** **1 = pose+scan** **2 = pose+reflectors+ scan**
PoseData		Pose data follow	UInt_16	**0 = no pose data** **1 = pose data follow**
Pose Data	X	X-Position	Int_32	**-10 000 000...** **+10 000 000 mm**
	Y	Y-Position	Int_32	**-10 000 000...** **+10 000 000 mm**

	Phi		Orientation	UInt_32	**0...360 000 mdeg**
	OptPoseData		Indicates if optional Pose data follow. Selected by showOpt-Param in NPOSPoseDataFormat	UInt_16	**0 = no optional data** **1 = optional data follow**
	Optional Pose Data	Output-Mode	Data are instantly or extrapolated	Enum_8	**0 = instantly** **1 = extrapolated**
		Time-stamp	Timestamp	UInt_32	**0...4 294 967 295 ms (0xffffffff)**
		Mean-Dev	Mean deviation	Int_32	**0... 2000 mm**
		Nav-Mode	Position mode	Enum_8	**0 = initial positioning** **1 = continuous positioning** **2 = virtual positioning** **3 = positioning stopped** **4 = position invalid** **5 = external**
		Info-State	Diagnosis Information	UInt_32	**reserved**
		QuantUsedReflectors	Number of used Reflectors	UInt_8	**0...40**
LandmarkData			Indicates if landmark data follow	UInt_16	**0 = no landmark data** **1 = landmark data follow**
Landmark Data	Landmark-Filter		Indicates the type of landmark data ; used, seen or expected	Enum_8	**0 = used** **1 = seen** **2 = expected landmarks**
	Reflectors		Number of landmark Data follow	UInt_16	**0...40**
	Reflector Data	Cart	Set to 1 if Cartesian Data follow. Selectable by format in NLMDLandmarkDataFormat	UInt_16	**0 = no Cartesian data** **1 = Cartesian data follow**
		Cart. X	X-Coordinate	Int_32	**-70 000...** **+70 000 mm**
		Y	Y-Coordinate	int_32	**-70 000...** **+70 000 mm**

		Polar		Set to 1 if polar Data follow. Selectable by format in NLMDLandmarkDataFormat	UInt_16	**0 = no polar data** **1 = polar data follow**
		Polar	Dist	Distance	UInt_32	**0... 70 000 mm**
			Phi	Orientation	UInt_32	**0...360 000 mdeg**
		optLandmarkData		Indicates if optional landmark data follow. Selected by showOptParam in NLMDLandmarkDataFormat	UInt_16	**0 = no optional data** **1 = optional landmark data follow**
		Optional Reflector Data	LocalID	Local ID	UInt_16	**0...40**
			GlobalID	Global ID	UInt_16	**0...11999**
			Type	Landmark type	Enum_8	**0 = undefined** **1 = reflector**
			Sub-type	Reflector type	Enum_16	**0 = undefined** **1 = flat** **2 = cylindric**
			Quality	Reserved	UInt_16	**reserved**
			Time-stamp	Timestamp	UInt_32	**0...4 294 967 295 ms (0xffffffff)**
			Size	Size	UInt_16	**1...150 mm (Default: 80mm)**
			HitCount	Number of hits	UInt_16	**1...1440**
			Mean-Echo	Mean Echo Amplitude	UInt_16	**0...1023**
			Index-Begin	Start Index of the reflector	UInt_16	**0...1439**
			Index-End	End Index of the reflector	UInt_16	**0...1439**
NAVscanData				Indicates the number of output channels of the Scan data; Channel with 32 bit values	UInt_16	**0 = no scan data** **1 = one output channel for scan data (distance)** **2 = two Output channels for scan data (distance + echo)**
NAVScan-Data	ContentType			Content of the channel.	String	**DIST1 : distance** **ANGL1 : angle**
	ScaleFactor			Multiplier	Real	**1 (for NAV350 always)**

	ScaleOffset		Offset	Real	**0 (for NAV350 always)**
	StartAngle		Start angle of the Scans	Int_32	**0...+360 000 mdeg**
	AngleRes		Angular step	UInt_16	**1/1000 degree ; 250 mdeg (fix)**
	TimestampStart		Timestamp Start of scan	UInt_32	**ms**
	Data		Number of scan data follow	UInt_16	**0...1440**
	Scan	AData	Scan data 1 to n. DIST in mm, ANGL in 1/10.000 degree	UInt_32	**00000000 ... FFFFFFFF**
NAVRemissionData			Indicates a 16 bit channel is enabled for remission data	UInt_16	**0 = no remission data 1 = one channel for remission data**
Remissions-Scan Data	ContentType		Type of data	String	**RSSI1 Remission**
	ScaleFactor		Multiplier	Real	**1 (for NAV350 always)**
	ScaleOffset		Offset	Real	**0 (for NAV350 always)**
	StartAngle		Start angle of the Scans	Int_32	**0...+360 000 mdeg**
	AngleRes		Angular step	UInt_16	**1/1000 degree ; 250 mdeg (fix)**
	TimestampStart		Timestamp at start of Scan	UInt_32	**0...4 294 967 295 ms (0xffffffff)**
	Data		Number of Scan data	UInt_16	**0...1 440**
	Scan	AData	Scan data 1 to n	UInt_16	**0000...FFFF**

2.7.3 Command: Velocity Input

By this command, the vehicle controller provides the current velocity vector to the NAV350 which is transformed from the AGV velocity vector.

Request

Command Syntax: **sMN mNPOSSetSpeed** *X Y Phi timestamp coordBase* (according: PV, Pv, Pw)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Set Velocity of the NAV350	String	**mNPOSSetSpeed**
X	X-component of velocity in the coordinate system defined by *coordBase*	Int_16	**-32 000 ...** **+32 000 mm/s**
Y	Y- component of velocity in the coordinate system defined by *coordBase*	Int_16	**-32 000 ...** **+32 000 mm/s**
Phi	Angular velocity of the NAV350	Int_32	**-360 000 ...** **+360 000 mdeg/s**
Timestamp	Timestamp of the Velocity vector related to the NAV350 clock	UInt_32	**0...4.294.967.295 ms (0xffffffff)**
CoordBase	Coordinate system of the velocity vector, local or global	Enum_8	**0 = local coordinate system of the NAV350** **1 = absolute coordinate system**

Response

Command Syntax: **sAN mNPOSSetSpeed** *errorCode*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method by name)	String	**sAN**
Command	Set Velocity of the NAV350	String	**mNPOSSetSpeed**
ErrorCode	Error code	Enum_8	**0 = no error** **1 = wrong operating mode** **3 = parameter out of range** **7 = general error**

2.7.4 Command: Set Current Position

This command sets initially the current position of the NAV350 by the vehicle controller.

Request

Command Syntax: **sMN mNPOSSetPose** *X Y Ph*

Command part	Description	Type	Range/Value
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Set current NAV350 position	String	**mNPOSSetPose**
X	X-Position of the NAV350	Int_32	**-10 000 000...** **+10 000 000 mm**
Y	Y-Position of the NAV350	Int_32	**-10 000 000...** **+10 000 000 mm**
Phi	Orientation of the NAV350	Int_32	**-360 000...** **+360 000 mdeg**

Response

Command Syntax: **sAN mNPOSSetPose** *errorCode*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method by name)	String	**sAN**
Command	Set current NAV350 position	String	**mNPOSSetPose**
ErrorCode	Error code	Enum_8	**0 = no error** **1 = wrong operating mode** **3 = parameter out of range** **7 = general error**

2.7.5 Command: Set Current Position by Landmark ID

This command provides a landmark ID of a visible landmark to find the initial current position.

Request

Command Syntax: **sMN mNPOSSetPoseID** *landmarkID* (according: PM)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS method by name)	String	**sMN**
Command	Set current NAV350 position	String	**mNPOSSetPoseID**
LandmarkID	Landmark ID in view of theNAV350	UInt_16	**0...11 999**

Response

Command Syntax: **sAN mNPOSSetPoseID** *errorCode*

Command part	Description	Type	Range/Value
Command type	Response (SOPAS method by name)	String	**sAN**
Command	Set current NAV350 position	String	**mNPOSSetPoseID**
ErrorCode	Error code	Enum_8	**0 = no error** **1 = wrong operating mode** **3 = parameter out of range** **7 = general error**

3 Result Port Protocol

3.1 Introduction of the Result Port Protocol

3.1.1 General

This document explains the usage of the additional result telegram.

The result port telegram has been introduced in the NAV3xx series starting from the firmware version V1.16.

The telegrams are optimized for a short transmission time and the data structure is are arranged for a memcopy that means to copy the data from the receiving buffer to the main memory for the further processing.

3.1.2 Usage of result port

For parametrization and request of data either the IP ports 2111 or 2112 could be used.
IP Port 2201 is used for unidirectional data output streaming.

Port 2111	SOPAS CoLaA-Protocol
Port 2112	SOPAS CoLaA/B-Protocol
Port 2201	ResultPort

The result port telegrams consist of a header structure followed by specific data. The payload type field indicates the transmitted result. Available are results of raw scan data, reflector detection and localization.

Each result could be enabled separately.

3.1.3 Result Port Payload Types

Result Payload Type	ID	Description
Scan Data	0x0101	Raw scan data (distance, angle, remission)
Reflector Detection	0x0601	Result of reflector detection (Available in Mode LMDetection)
Localization	0x0641	Result of localization (Available in Mode Navigation)
Scandata (Little Endian)	0x0181	Raw scan data in little endian format
Reflector Detection (Little Endian)	0x0681	Result of reflector detection in little endian format (Available in Mode LMDetection)
Localization (Little Endian)	0x06c1	Result of localization in little endian format (Available in Mode Navigation)

These combinations are possible (either in Big Endian or Little Endian Format) :

LANDMARK Detection Mode

- Scan Data only
- Reflector Detection only
- Scan Data + Reflector Detection

NAVIGATION Mode

- Scan Data only
- Localization only
- Scan Data + Localization

3.2 Framing

The Result Protocol consists of an unique header structure and telegram specific content. The telegram content is determined by the field *PayloadType*.

Section	Fields		Description	Type	Size [Byte]	Total Size
HEADER	MagicWord	"SICK"	Magic word (0x53 0x49 0x43 0x4B)	4x UInt8	4	
	Length		Length of telegram. Header, payload and Trailer.	UInt32	4	
	PayloadType		Payload datatype (see table 2.2)	UInt16	2	
	PayloadVersion		Version of PayloadType structure.	UInt16	2	
	OrderNumber		Order number of device	UInt32	4	
	SerialNumber		Serial of device	UInt32	4	
	FW-Version		Softwareversion of device	20x UInt8	20	
	TelegramCounter		Telegram counter since power on.	UInt32	4	
	SystemTime		System time of related scan data.	NTP	8	52

Section	Fields		Description	Type	Size [Byte]	Total Size
Trailer	Checksum		CRC16 over Length without 2 bytes of trailer	UInt16	2	2

2.1.3. Checksum Calculation

Current implementation is using the standard EDP1 / EDP2 CRC16 calculation.

CRC-CCITT

Polynom 0x1021 (without leading 1: x16 + x12 + x5 + 1)

Start value 0xffff

3.2.1 Result of Landmark Detection

By parameter *ER1FctLMDetectFixedLength* the number of transmitted landmarks is either variable according to the number of seen landmarks or fixed.

Parameter *ER1FctLMDetectMaxLength* defines the maximum number of transmitted landmarks.

If enabling *ER1FctLMDetectFixedLength,* empty landmark positions will be filled with a zero value up to *ER1FctLMDetectMaxLength.*

Section	Fields		Description	Type	Size [Byte]	Total Size
LANDMARKS	ErrorCode		0: OK 1: UNKNOWNERROR	UInt16	2	
	ScanCounter		Counter of related scan data	UInt32	4	
	Content		Flag pattern: 0x01=Output with fixed length	UInt32	4	
	LandmarkNum		Number of landmark data	UInt16	2	
	0 – 60 1. Maximum length is configurable. Default is 40. 2. Telegram is filled with zero, if number of seen landmarks is smaller than maximum length.	Timestamp	Timestamp of Reflector	UInt32	4	
		X	Cartesian coordinates	Int32	4	
		Y		Int32	4	
		Distance	Polar coordinates	UInt32	4	
		Angle		Int32	4	
		Type	Type of Landmark	UInt16	2	
		ID	Global ID of Landmark in Navigation Mode	UInt32	4	
		Reserved		UInt32	4	
		Size	Size	UInt16	2	
		Hit count	Number of hits	UInt16	2	
		RSSI	Mean echo amplitude	UInt16	2	
		Reserved		UInt32	4	
		Index begin	Start index of the landmark in related scandata	UInt16	2	
		Index end	End index	UInt16	2	16+N*44 = **2656**

3.2.2 Result of Localization

Section	Fields	Description	Type	Size [Byte]	Total Size
POSITION	ErrorCode	0: OK 1: UNKNOWNERROR	UInt16	2	
	ScanCounter	Counter of related scan data	UInt32	4	
	Timestamp	Timestamp of position	UInt32	4	
	X	Cartesian global coordinates	Int32	4	
	Y		Int32	4	
	Orientation	Orientation in global system	Int32	4	
	MeanDeviation	Mean Deviation of position	Int32	4	
	Properties	Flag pattern: reserved	UInt16	2	
	NavMode	Mode/State of localization core 0: INITIAL 1: CONTINUOUS 2: VIRTUAL (Movement is extrapolated) 3: STOP 4: INVALID (No valid position) 5: EXTERNAL (External input of position used)	UInt16	2	
	InfoState	Flag pattern. Shows info about used algorithm	UInt32	4	
	NumUsedRefl	Number of used reflectors for this position	UInt16	2	
	Reserved		UInt32	4	
	Reserved		UInt32	4	44

3.2.3 Scandata Result

Section	Fields		Description	Type	Size [Byte]	Total Size
Header	ErrorCode		0: OK 1: UNKNOWNERROR	UInt16	2	
	ScanCounter		Number of scans	UInt32	4	
	Timestamp		Timestamp of Scandata	UInt32	4	
	DeviceState		Device status	UInt16	2	
	ScanFreq		Scan frequency	UInt32	4	16
	ChannelNum	32bit channnels	Amount of 32Bit channels	Uint16	2	2
32 bit Channel	Channel Header	Content	Content: Dist1; Angl1	6x UInt8	6	
		ScaleFactor	Scale factor	Float32	4	
		ScaleOffset	Scale factor offset	Float32	4	
		StartAngle	Start angle of scan (1/10.000°)	UInt32	4	
		Steps	Angle res. (1/10.000°)	UInt16	2	
		ScanPoints	Number of scan points	UInt16	2	22
	1440x	ScanData	Data : Distance or Angle with 32bit	Int32	4	5760
	ChannelNum	16 bit channels	Amount of 16Bit channels	UInt16	2	2
16bit Channel	Channel Header	Content	Content: RSSI1	6x UInt8	6	
		ScaleFactor	Scale factor	Float32	4	
		ScaleOffset	Scale factor offset	Float32	4	
		StartAngle	Start angle of scan (1/10.000°)	UInt32	4	
		Steps	Angle res. (1/10.000°)	UInt16	2	
		ScanPoints	Number of scan points	UInt16	2	22
	1440x	ScanData	Point echo	Int16	2	2880
						8704

3.3 Parametrization and Visualization

3.3.1 Parametrization

Parametrization is done in SOPAS CoLa protocol or SOPAS ET GUI. The Result Port itself is mentioned as unidirectional data output stream.

Available Parameters:

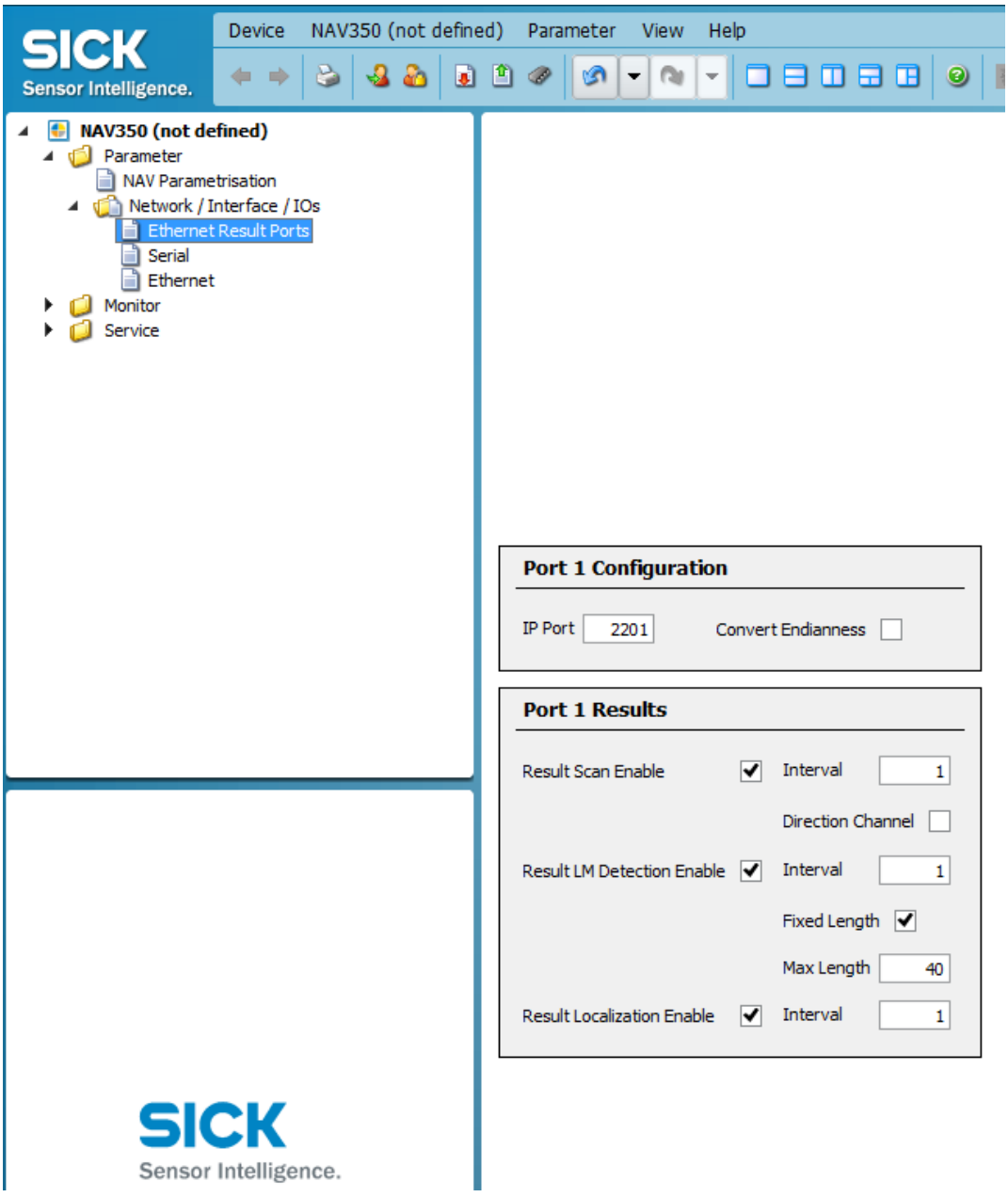
Name	Type	Description
RS1Port	UInt16 (default 2201)	Port for TCP/IP communication of EtherResult port 1
ER1Request	UInt16 (default 0xffff)	Requests specific number for all activated results on EtherResult port 1. System decrements parameter for each result processing cycle independently from the specific Interval parameter. It is based on internal scan data processing. Default value of 0xffff means unlimited number of results.
ER1RequestConvertEndianness	Bool (default false)	Switches all enabled results to their converted equivalent.
ER1FctScanEn	Bool (default false)	Activation of Scan Data EtherResult Port 1
ER1FctScanInterval	UInt16 (default 1)	Interval for output of result messages (1 = each scan is sent out, 2 = each second scan is sent out, ...)
ER1FctScanDirChannel	Bool (default false)	Enables output of direction channel.
ER1FctLMDetectEn	Bool (default false)	Activation of Landmark Detection EtherResult Port 1
ER1FctLMDetectInterval	UInt16 (default 1)	Interval for output of result messages (1 = each scan is sent out, 2 = each second scan is sent out, ...)
ER1FctLMDetectFixedLength	Bool (default true)	A fixed number of landmarks are transmitted. If fewer landmarks detected, zero values are transmitted.
ER1FctLMDetectMaxLength	UInt16 (default 40, max 60)	Limitation of transmitted landmark number.
ER1FctLocalizationEn	Bool (default false)	Activation of Localization EtherResult Port 1
ER1FctLocalizationInterval	UInt16 (default 1)	Interval for output of result messages (1 = each scan is sent out, 2 = each second scan is sent out, ...)

3.4 Visualization in SOPAS

The configuration could be done by Cola Command or by SOPAS.

In the menu Ethernet Result port are the options to select the format to Big Endian (MSB first / default) or Little Endian (LSB first) by “Convert endianness” and to select the output of Scan data and / or Landmark data

If the output of the Landmark configuration data is selected the Landmark Dataformat maybe defined the maximum number of landmarks that will be given out.

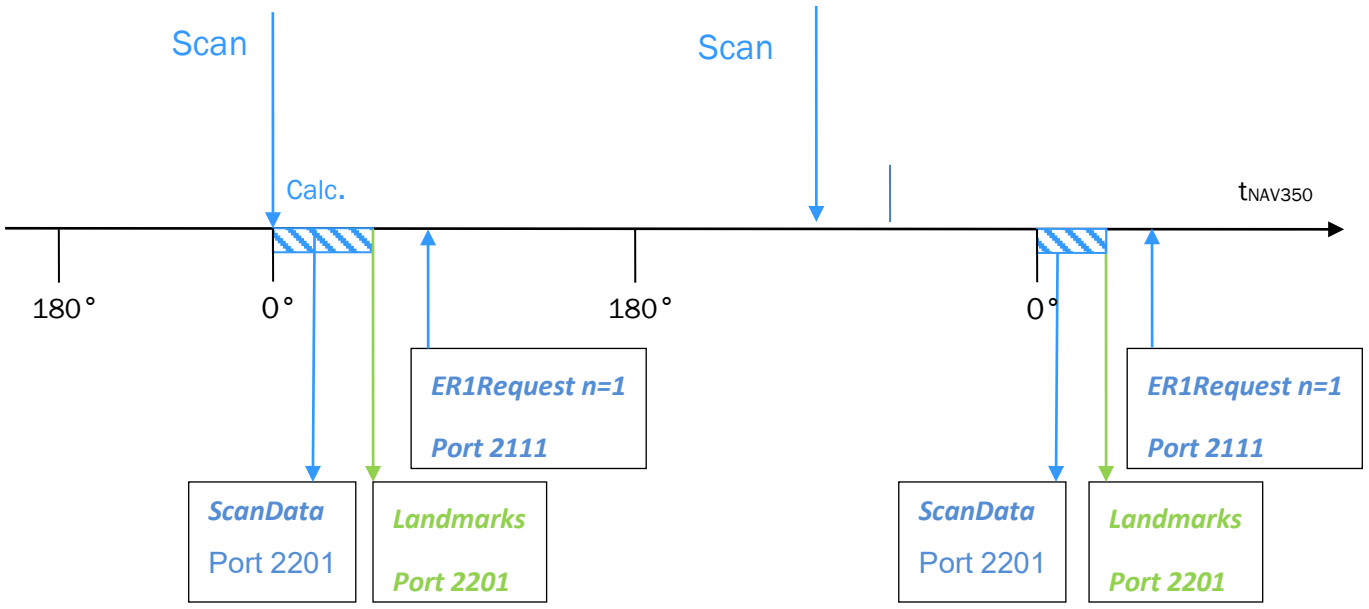


3.5 Example

Activation of the results for raw scan data and landmark detection by setting *ER1FctScanEn* and *ER1FctLMDetectEn* to true.

By sending *ER1Request n* via IP port 2111, the scanner will transmit n data results after processing via port 2201.

The scanner will transmit landmark data and scan data approximately every 125 ms.



Remark :

The sequence of the transmission may differ.
The content of the data transmission is indicated by the ID of the payload type.

813892/1A6R/2022-02-22 · 8M/TW · A4 4c int49

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

SICK
Sensor Intelligence.