

# Telegrams for Configuring and Operating the NAV350 Laser Positioning Sensor



Navigating the route to improved productivity



**Copyright**

Copyright © 2010 - 2013

SICK AG Waldkirch

Identification & Measuring, Reute Plant

Nimburger Strasse 11

79276 Reute

Germany

**Trademark**

Acrobat™ Reader™ is a trademark of Adobe Systems Incorporated.

Telegram listing version

For the latest version of this telegram listing (PDF), see [www.sick.com](http://www.sick.com).

---

<b>1</b>	<b>INTRODUCTION</b>	<b>4</b>
1.1	Function	4
1.2	Command Formats	4
1.2.1	Protocols in SOPAS ET	4
1.2.2	Binary Protocol	4
1.2.3	ASCII Protocol	5
1.2.4	Summary Example and Format Explanation	6
1.3	Data Types of Variables	7
1.4	Storable Parameter	8
1.5	Initialization Sequence	10
1.6	Sequence for Reflector Mapping	10
1.7	Principle of Navigation Mode	11
1.7.1	Navigational Loop	11
1.7.2	Providing Speed Loop	11
1.8	Timing Set Speed	12
1.8.1	Synchronization	12
1.8.2	Interpretation Timestamp/Position	13
1.8.3	Interpretation Timestamp / Speed with Synchronized AGV Timer (recommended)	14
1.8.4	Timing NPOSSetSpeed Based on Constant Offset	14
1.9	Error Messages (sFA)	15
<b>2</b>	<b>COMMANDS</b>	<b>16</b>
2.1	Commands Accessible from all Modes	16
2.1.1	Command: Read Device Identification	16
2.1.2	Command: Read Serial Number	17
2.1.3	Command: Read Measurement Firmware Version	17
2.1.4	Command: Read Application Version	18
2.1.5	Command: Set Current Layer	18
2.1.6	Command: Define Reflector Identification Window	20
2.1.7	Command: Configure Mapping	22
2.1.8	Command: Set the Sliding Mean	24
2.1.9	Command: Set Positioning Data Format	25
2.1.10	Command: Set Landmark Data Format	26
2.1.11	Command: Set Scan Data Format	28
2.1.12	Command: Set Hardware Time Sync	29
2.1.13	Command: Set Reflector Size	31
2.1.14	Command: Set Reflector Type	32
2.1.15	Command: Set Landmark Matching	33
2.1.16	Command: Set Sector Muting	34
2.1.17	Command: Set Orientation Coordinate System	36
2.1.18	Command: Set N Closest Reflectors	37
2.1.19	Command: Set Action Radius	38
2.1.20	Command: Set Reflector Threshold	39
2.2	Method Call	41
2.3	Methods Accessible from all Modes except POWERDOWN	41
2.3.1	Command: Set Operating Mode	41

---

---

2.3.2	Command: Set User Level	43
2.3.3	Command: Store Data Permanent	44
2.3.4	Command: Synchronize Timestamp	45
2.3.5	Command: Break Asynchronous Method	46
2.3.6	Command: Device Reset	47
2.4	Methods in STANDBY Mode	48
2.4.1	Command: Serial Interface Configuration	48
2.4.2	Command: Change IP Configuration	50
2.4.3	Command: Change Ethernet Configuration	51
2.4.4	Command: DHCP Enabled / Disabled	52
2.4.5	Command Add Landmark	53
2.4.6	Command: Edit Landmark	55
2.4.7	Command: Delete Landmark	56
2.4.8	Command: Read Landmark	57
2.4.9	Command : Read Layer	58
2.4.10	Command: Read Layout	59
2.4.11	Command: Erase Layout	60
2.4.12	Command: Store Layout Permanent	61
2.5	Methods in MAPPING Mode	62
2.5.1	Command: Do Mapping	62
2.6	Methods in LANDMARK Mode	65
2.6.1	Command: Get Landmark data	65
2.7	Methods in NAVIGATION Mode	69
2.7.1	Command: Position Request	69
2.7.2	Command: Position Data Request	73
2.7.3	Command: Velocity Input	78
2.7.4	Command: Set Current Position	79
2.7.5	Command: Set Current Position by Landmark ID	80

---

# 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 (**Command Language**) are described. There are two types of the **Command Language**: 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:**

```
02734D4E205365744163636573734D6F6465203320463437323437343403
```

(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.45)) 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	<b>sMN</b>	<b>73 4D 4E</b>
Command	User level	String	<b>SetAccessMode</b>	<b>53 65 74 41 63 63 65 73 73 4D 6F 64 65</b>
User level	Select user level	Int_8	<b>2 = maintenance 3 = authorized client</b>	<b>02 = maintenance 03 = authorized client</b>
Password	“Hash” - value for the user level “Maintenance” “Authorized Client”	UInt_32	<b>B21ACE26 F4724744</b>	<b>B2 1A CE 26 F4 72 47 44</b>

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 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	$\pm \sim 10^{-44,85} \dots +10^{38,53}$	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 hexadecimal 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
<b>Serial configuration</b>			
Baudrate	SIHstBaud	<b>6</b> <b>7</b> <b>8</b> <b>9</b>	<b>19200</b> <b>38400</b> <b>57600</b> <b>115200 (Default)</b>
Data bits	Data Bits	<b>7..8</b>	<b>7_data bits</b> <b>8_data bits (Default)</b>
Parity	Parity Check	<b>0..4</b>	<b>0 = none</b> <b>1 = odd</b> <b>2 = even (Default)</b> <b>3 = mark</b> <b>4 = space</b>
Stopbit	SIHstStop	<b>0</b> <b>1</b>	<b>1 (Default)</b> <b>2</b>
Device ID	NetDeviceID	<b>1 ... 127</b>	<b>(Default: 1)</b>
<b>Ethernet configuration</b>			
IP-Address	EtherIPAddress	<b>0 ... 255 (4x)</b>	<b>192.168.1.10 (Default)</b>
Gateway	EtherIPGateAddress	<b>0 ... 255 (4x)</b>	<b>0.0.0.0 (Default)</b>
Subnetmask	EtherIPMask	<b>0 ... 255 (4x)</b>	<b>255.255.255.0 (Default)</b>
Interfaced speed and duplex mode	EtherIPSpeedDuplex	<b>0</b> <b>1</b> <b>2</b> <b>3</b> <b>4</b>	<b>auto (Default)</b> <b>10MB half</b> <b>10MB full</b> <b>100MB half</b> <b>100MB full</b>
Addressmode (DHCP)	EtherAddressingMode	<b>0</b> <b>1</b>	<b>disabled (Default)</b> <b>enabled</b>

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

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	<b>sMN SetAccessMode</b>	43	<i>newMode</i> <i>password</i>	3 F4724744
2	Standby mode	<b>sMN mNEVAChangeState</b>	41	<i>mode</i>	<b>1</b>
3	Write current Layer	<b>sWN NEVACurrLayer</b>	18	<i>currLayer</i>	<b>7</b>
4	Set positioning data format	<b>sWN NPOSPOSEDataFormat</b>	25	<i>outputMode</i>	<b>1 0</b>
5	Navigation mode	<b>sMN mNEVAChangeState</b>	41	<i>mode</i>	<b>4</b>
6	Requesting the position	<b>sMN mNPOSGetPose</b>	69	<i>wait</i>	<b>1</b>

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.

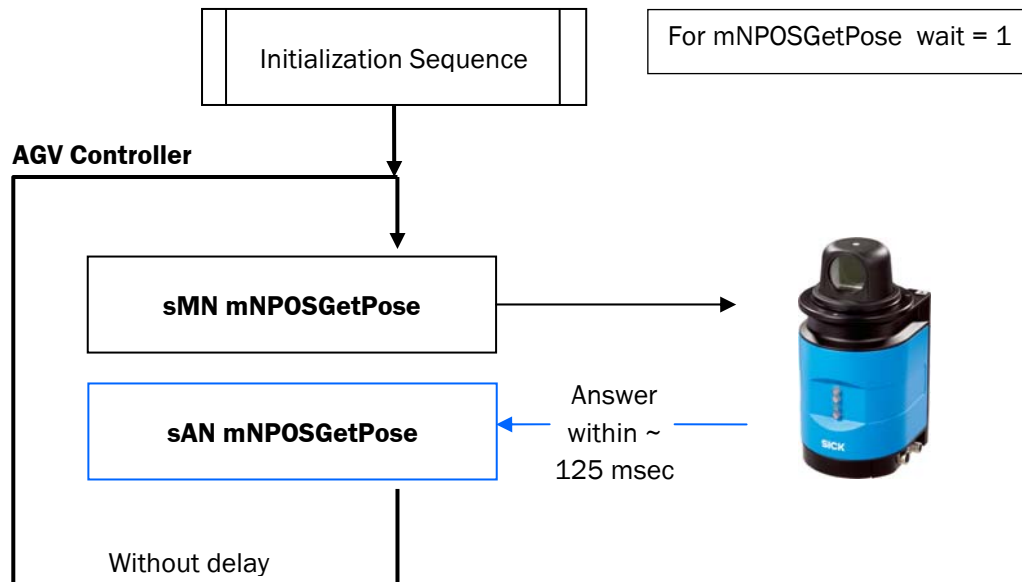
	Step	Command	Page	Parameter
1	Configure mapping (Settings for the mapping)	<b>sWN NMAPMapCfg</b>	22	<i>mean</i> <i>negative</i> <i>x y phi</i>
2	Select the layer number	<b>sWN NEVACurrLayer</b>	20	<i>Current layer</i>
2	Do mapping (Performs mapping and sends the measured reflectors data to PC/PLC)	<b>sMN mNMAPDoMapping</b>	70	
3	Add landmark (Writes 1..50 landmarks to the scanner)	<b>sMN mNLAYAddLandmark</b>	59	<i>landmarkData</i>
4	Store layout permanent (Saves the reflectors in the flash memory)	<b>sMN mNLAYStoreLayout</b>	65	

Tabelle 1-7: Sequence for reflector mapping

## 1.7 Principle of Navigation Mode

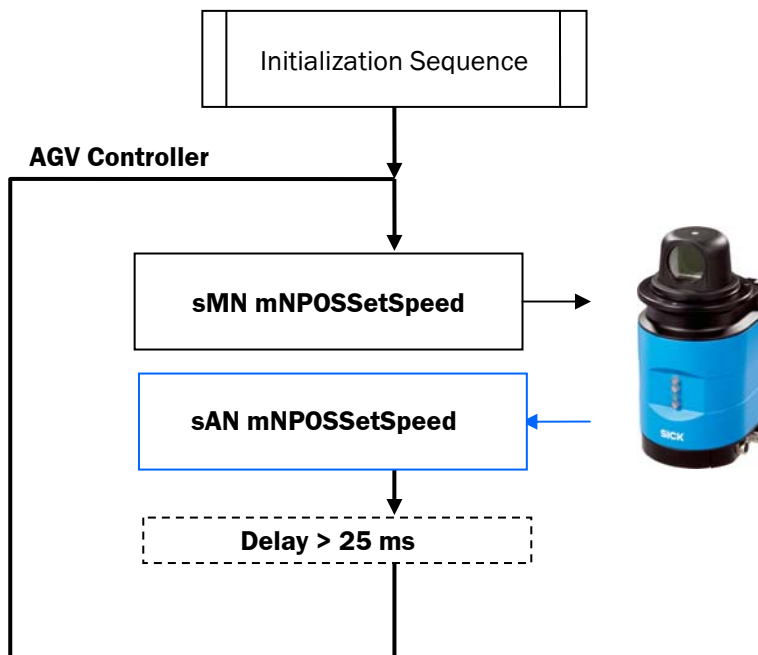
This chapter shows the sequences from the initialization to the navigational loop to request the current position from the scanner.

### 1.7.1 Navigational Loop



### 1.7.2 Providing Speed Loop

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

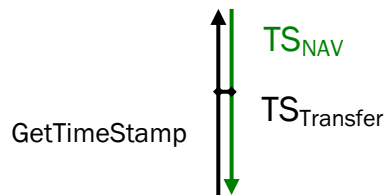


## 1.8 Timing Set Speed

### 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.

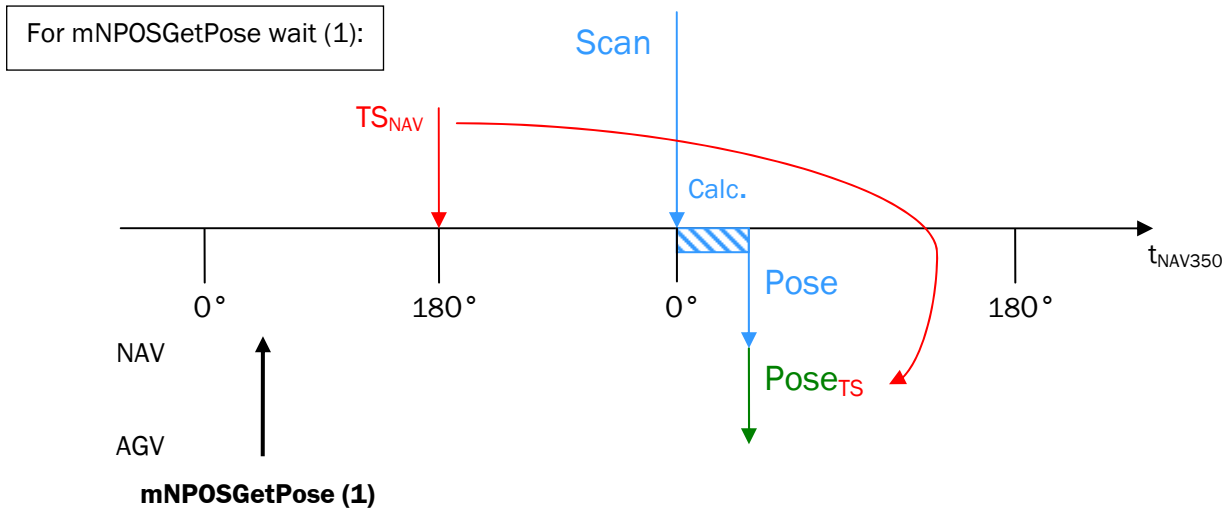
GetTimeStamp → 
$$TS_{\text{offset}} = (TS_{\text{AGV}} - TS_{\text{Transfer}}) - TS_{\text{NAV}}$$



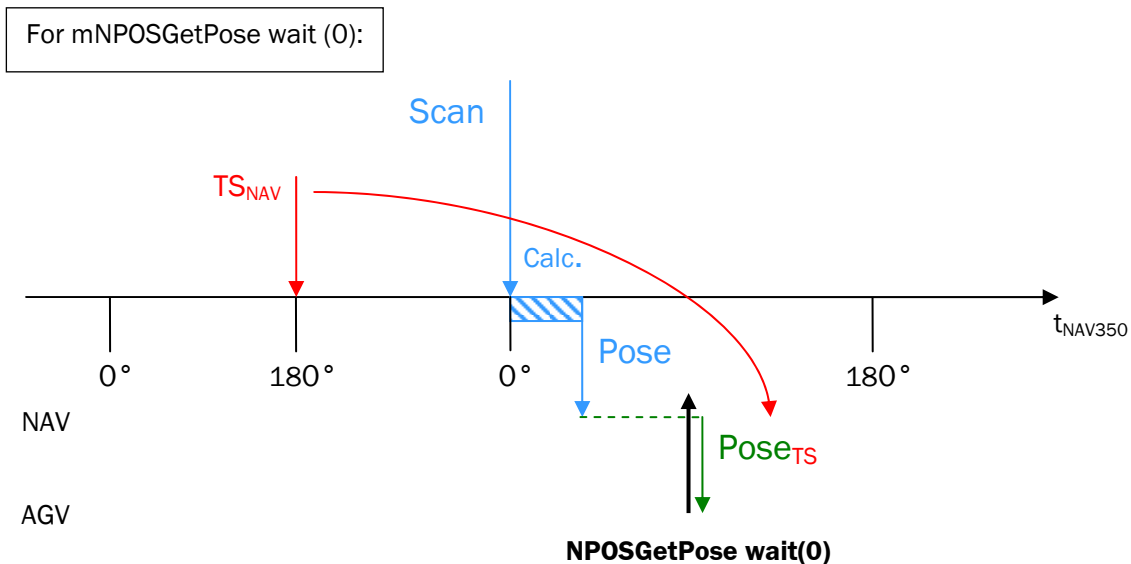
$TS_{\text{offset}}$	Resulting timing offset
$TS_{\text{AGV}}$	Timestamp of AGV
$TS_{\text{NAV}}$	Timestamp of NAV350
$TS_{\text{Transfer}}$	Transfer time via RS232 or Ethernet interface

### 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 ( $POSE_{TS}$ ).

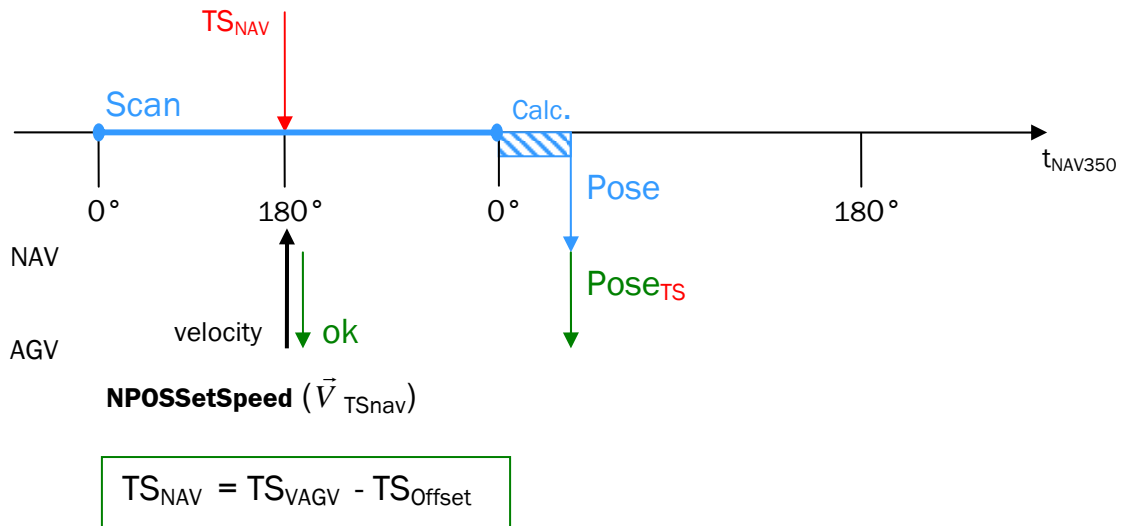


The command `mNPOSGetPose wait (0)` requests the last available position information of the NAV350 and gets the respond immediately with the timestamp (TS) from the previous 180° angle position.



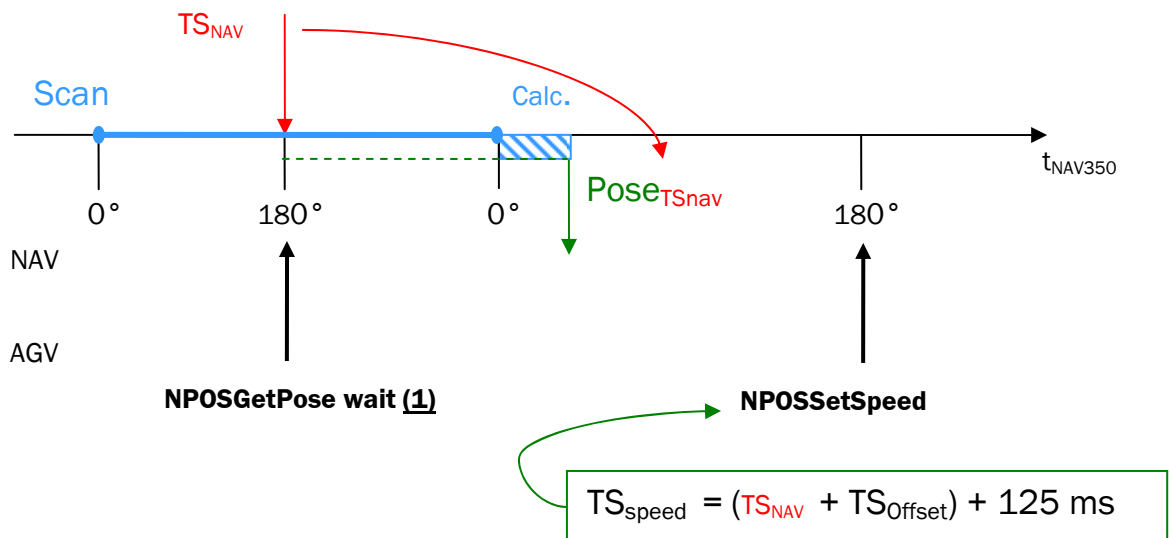
### 1.8.3 Interpretation Timestamp / Speed with Synchronized AGV Timer (recommended)

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



### 1.8.4 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  $180^\circ$  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 Devicelident** (according: ST)

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	<b>sRN</b>
Command	Read Device identification	String	<b>Devicelident</b>

##### Response

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

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	<b>sRA</b>
Command	Read Device identification	String	<b>Devicelident</b>
SizeName	Length of name string	UInt_8	<b>0 ... 255</b>
Name	Device name	String	...
SizeVersion	Length of version string	UInt_8	<b>0 ... 255</b>
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	<b>sRN</b>
Command	Read serial number	String	<b>SerialNumber</b>

#### Response

Command Syntax: **sRA SerialNumber sizeNumber serialNumber**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	<b>sRA</b>
Command	Read Serial number	String	<b>SerialNumber</b>
SizeNumber	Length serial number string	UInt_8	<b>0 ... 16</b>
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	<b>sRN</b>
Command	Read Measurement firmware Version	String	<b>MMDeviceInfo</b>

#### Response

Command Syntax: **sRA MMDeviceInfo version**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	<b>sRA</b>
Command	Read Measurement firmware Version	String	<b>MMDeviceInfo</b>
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	<b>sRN</b>
Command	Read Application software version	String	<b>FirmwareVersion</b>

#### Response

Command Syntax: **sRA FirmwareVersion sizeVersion version**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	<b>sRA</b>
Command	Application software version	String	<b>FirmwareVersion</b>
SizeVersion	Length of Application version string	UInt_8	<b>0 ... 255</b>
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	<b>sWN</b>
Command	Set current Layer	String	<b>NEVACurrLayer</b>
CurrLayer	Layer number	UInt_16	<b>0 ... 319</b> <b>(Default: 0)</b>

**Response**Command Syntax: **sWA NEVACurrLayer**

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

**Request Read**Command Syntax: **sRN NEVACurrLayer**

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

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

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

## 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	<b>sWN</b>
Command	Define Reflector Identification Window	String	<b>NCORIdentWindow</b>
WinLow	Window radius in mm at <i>distLow</i>	UInt_16	<b>100 ... 2000 mm</b> <b>(Default: 300 mm)</b>
WinHigh	Window radius in mm at <i>distHigh</i>	UInt_16	<b>100 ... 2000 mm</b> <b>(Default: 300 mm)</b>
DistLow	Distance in mm at <i>winLow</i>	UInt_32	<b>500 ... 70 000 mm</b> <b>(Default: 500 mm)</b>
DistHigh	Distance in mm at <i>winHigh</i>	UInt_32	<b>500 ... 70 000 mm</b> <b>(Default: 70 000 mm)</b>

**Response**Command Syntax: **sWA NCORIdentWindow**

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

**Request Read**Command Syntax: **sRN NCORIdentWindow**

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

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

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

### 2.1.7 Command: Configure Mapping

The command "Configure Mapping" sets the parameter to control the Mapping mode (refer command *Do Mapping* (p.62))

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.18)), 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.31)).

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	<b>sWN</b>
Command	Configure Mapping	String	<b>NMAPMapCfg</b>
Mean	Number of scans for averaging	UInt_8	<b>1...127</b> <b>(Default: 50)</b>
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	<b>0 = positive-Mapping (Default)</b>  <b>1 = negative-Mapping</b>
X	X-Position of the NAV350	Int_32	<b>-10 000 000...</b> <b>+10 000 000 mm</b> <b>(Default: 0)</b>
Y	Y-Position of the NAV350	Int_32	<b>-10 000 000...</b> <b>+10 000 000 mm</b> <b>(Default: 0)</b>
Phi	Heading of the NAV350	Int_32	<b>-360 000...</b> <b>+360 000 mdeg</b> <b>(Default: 0)</b>

**Response**Command Syntax: **sWA NMAPMapCfg**

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

**Request Read**Command Syntax: **sRN NMAPMapCfg**

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

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

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	<b>sRA</b>
Command	Read Mapping configuration	String	<b>NMAPMapCfg</b>
Mean	Number of scans for averaging	UInt_8	<b>1...127</b>
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	<b>0 positive-Mapping 1 negative-Mapping</b>
X	X-Position of the NAV350	Int_32	<b>-10 000 000... +10 000 000 mm</b>
Y	Y-Position of the NAV350	Int_32	<b>-10 000 000... +10 000 000 mm</b>
Phi	Orientation of the NAV350	Int_32	<b>-360 000... +360 000 mdeg</b>



### 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	<b>sWN</b>
Command	Set sliding mean	String	<b>NPOSSlidingMean</b>
SlidingMean	Smoothing factor	UInt_8	<b>1 ... 63</b> <b>(Default: 1)</b>

#### Response

Command Syntax: **sWA NPOSSlidingMean**

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

#### Request Read

Command Syntax: **sRN NPOSSlidingMean**

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

#### Response

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

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

### 2.1.9 Command: Set Positioning Data Format

The command Set Positioning Data Format parameterizes the responds of the commands *NPOSGetData* (p.73) and *NPOSGetPose* (p.69).

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.73)**)

#### Request Write

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

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

#### Response

Command Syntax: **sWA NPOSPoseDataFormat**

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

#### Request Read

Command Syntax: **sRN NPOSPoseDataFormat**

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

**Response**

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

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

**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.65) and method *mNPOSGetData* (p.73). 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	<b>sWN</b>
Command	Set landmark data format	String	<b>NLMDLandmarkDataFormat</b>
Format	Select Cartesian or polar	Enum_8	<b>0 = Cartesian (Default)</b> <b>1 = polar</b>
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	<b>0 = suppress (Default)</b> <b>1 = enable</b>
LandmarkFilter	Select transmission of the optional landmark data in positioning mode with <i>NPOSGetData</i>	Enum_8	<b>0 = used</b> <b>1 = detected (Default)</b> <b>2 = expected landmark</b>

**Response**Command Syntax: **sWA NLMDLandmarkDataFormat**

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

**Request Read**Command Syntax: **sRN NLMDLandmarkDataFormat**

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

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

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	<b>sRA</b>
Command	Read landmark data format	String	<b>NLMDLandmarkDataFormat</b>
Format	Response Cartesian or polar	Enum_8	<b>0 = Cartesian</b> <b>1 = polar</b>
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	<b>0 = suppress</b> <b>1 = enable</b>
LandmarkFilter	Select transmission of the optional landmark data in positioning mode with <i>NPOSGetData</i>	Enum_8	<b>0 = used</b> <b>1 = detected</b> <b>2 = expected landmark</b>

### 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.65) and method *mNPOSGetData* (p.73). 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	<b>sWN</b>
Command	Set scan data format	String	<b>NAVScanDataFormat</b>
DataMode	Select distance and/or angle	Enum_8	<b>0 = none</b> <b>1 = distance only (Default)</b> <b>2 = distance and angle</b>
ShowRSSI	Set transmission of remission data (echo)	Bool_1	<b>0 = suppress (Default)</b> <b>1 = enable</b>

#### Response

Command Syntax: **sWA NAVScanDataFormat**

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

#### Request Read

Command Syntax: **sRN NAVScanDataFormat**

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

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

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

**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.45) 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	<b>sWN</b>
Command	Set Hardware Time Sync	String	<b>NAVHardwareTimeSync</b>
Mode	Setting cycling pulse or on command synchronize timestamp	Enum_8	<b>0 = cyclic pulse</b> <b>1 = on request (Default)</b>
Mask	Cycle time in bit (modulo base 2)	UInt_8	<b>10...20 bit</b> <b>(Default: 15 bit)</b>

**Response**Command Syntax: **sWA NAVHardwareTimeSync**

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

**Request Read**Command Syntax: **sRN NAVHardwareTimeSync**

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

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

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

### 2.1.13 Command: Set Reflector Size

This command parameterizes the reflector size.

#### Request Write

Command syntax: **sWN NLMDRefISize** *size*

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

#### Response

Command syntax: **sWA NLMDRefISize**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	<b>sWA</b>
Command	Set reflector size	String	<b>NLMDRefISize</b>

#### Request Read

Command syntax: **sRN NLMDRefISize**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	<b>sRN</b>
Command	Read reflector size	String	<b>NLMDRefISize</b>

#### Response

Command syntax: **sRA NLMDRefISize** *size*

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



### 2.1.14 Command: Set Reflector Type

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

#### Request Write

Command syntax: **sWN NLMDRefIType** *type*

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

#### Response

Command syntax: **sWA NLMDRefIType**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS write by name)	String	<b>sWA</b>
Command	Set reflector type	String	<b>NLMDRefIType</b>

#### Request Read

Command syntax: **sRN NLMDRefIType**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	<b>sRN</b>
Command	Get reflector type	String	<b>NLMDRefIType</b>

#### Response

Command syntax: **sRA NLMDRefIType** *type*

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

### 2.1.15 Command: Set Landmark Matching

In this command you can set the rules for detecting reflectors. Detection 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	<b>sWN</b>
Command	Set reflector type	String	<b>NLMDLandmarkMatching</b>
Filter	Choose between the acceptance of only optimal detected reflectors (0), optimal and from a flat angle detected reflectors (1) or the two first options plus partly covered reflectors (2).  The given specifications are only valid for the "optimal" setting	Enum_8	<b>0 = optimal (Default)</b> <b>1 = optimal + angle</b> <b>2 = optimal + angle + partly covered</b>

#### Response

Command syntax: **sWA NLMDLandmarkMatching**

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

#### Request Read

Command syntax: **sRN NLMDLandmarkMatching**

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

**Response**

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

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

**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	<b>sWN</b>
Command	Configuration of sector muting	String	<b>NLMDMutedSectors</b>
<b>4 x sector</b>	AngleFrom	Start angle of the sector	UInt_32 <b>0 ... 359 999 mdeg (Default: 0)</b>
	AngleTo	End angle of the sector	UInt_32 <b>0 ... 359 999 mdeg (Default: 0)</b>
	Active	Activation of the sector	Bool_1 <b>0 = inactive (Default)</b> <b>1 = active</b>

**Response**

Command syntax: **sWA NLMDMutedSectors**

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

**Request Read**Command syntax: **sRN NLMDMutedSectors**

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

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

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

### 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	<b>sWN</b>
Command	Set rotate direction relative to global coordinate system	String	<b>NEVACoordOrientation</b>
Direction	Set orientation of the coordinate system	Enum_8	<b>0 = clockwise</b> <b>1 = counterclockwise (Default)</b>

#### Response

Command syntax: **sWA NEVACoordOrientation**

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

#### Request Read

Command syntax: **sRN NEVACoordOrientation**

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

#### Response

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

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

### 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	<b>sWN</b>
Command	Set N closest reflectors	String	<b>NLMDnClosest</b>
NClosest	Quantity of N closest reflectors	UInt_8	<b>0 ... 40</b> <b>(Default: 0)</b>

#### Response

Command syntax: **sWA NLMDnClosest**

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

#### Request Read

Command syntax: **sRN NLMDnClosest**

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

#### Response

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

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

### 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	<b>sWN</b>
Command	Set Action Radius	String	<b>NLMDActionRadius</b>
RFr	Minimum Actions Radius	UInt_32	<b>400 ... 70 000 mm (Default: 500)</b>
RTo	Maximum Action Radius	UInt_32	<b>400 ... 70 100 mm (Default: 70 000)</b>

#### Response

Command syntax: **sWA NLMDActionRadius**

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

#### Request Read

Command syntax: **sRN NLMDActionRadius**

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

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

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

**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 NLMDRefIThreshold** *percent* (according: RP, SP)

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

**Response**Command syntax: **sWA NLMDRefIThreshold**

Command part	Description	Type	Range/Value
Command type	Response (SOPAS read by name)	String	<b>sWA</b>
Command	Set reflector threshold	String	<b>NLMDRefIThreshold</b>



**Request Read**Command syntax: **sRN NLMDRefIThreshold**

Command part	Description	Type	Range/Value
Command type	Request (SOPAS read by name)	String	<b>sRN</b>
Command	Request Reflector Threshold	String	<b>NLMDRefIThreshold</b>

**Response**Command syntax: **sRA NLMDRefIThreshold** *percent*

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

## 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 acknowledgement 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 mNEVChangeState with "sMA" followed by the response "sAN mNEVChangeState" 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 mNEVChangeState mode** (according: SA, DA, UA, RA, MA, PA, PN)

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

**Response (Method Acknowledge)**Command Syntax: **sMA mNEVAChangeState**

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

**Response after Execution**Command Syntax: **sAN mNEVAChangeState** *errorCode mode*

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

### 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	<b>B21ACE26</b>
Authorized client	client	<b>F4724744</b>

#### Request

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

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

#### Response

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

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

### 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	<b>sMN</b>
Command	Store data permanent	String	<b>mEEwriteall</b>

#### Response

Command Syntax: **sAN mEEwriteall success**

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

### 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	<b>sMN</b>
Command	Sync Timestamp	String	<b>mNAVGetTimestamp</b>

#### Response

Command Syntax: **sAN mNAVGetTimestamp errorCode timestamp**

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

### 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	<b>sMN</b>
Command	Break Asynchronous Method	String	<b>mNAVBreak</b>

#### Response

Command Syntax: **sAN mNAVBreak errorCode count**

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

### 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	<b>sMN</b>
Command	Device reset	String	<b>mNAVReset</b>

#### Response

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

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



## 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	<b>sMN</b>
Command	Serial configuration	String	<b>mChangeSerialCfg</b>
Baudrate	Baudrate	Enum_8	<b>0 = 19200</b> <b>1 = 38400</b> <b>2 = 57600</b> <b>3 = 115200 (Default)</b>
Data bits	Number of data bits in each character	UInt_8	<b>7 = 7 Data bits</b> <b>8 = 8 Data bits (Default)</b>
Parity	Parity check	Enum_8	<b>0 = none</b> <b>1 = odd</b> <b>2 = even (Default)</b> <b>3 = mark</b> <b>4 = space</b>
Stopbits	Stopbits	UInt_8	<b>1 (Default)</b>

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

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

## 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.52). To activate the settings, it is necessary to run a device reset (*sMN mNAVReset* (p.47)). 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	<b>sMN</b>
Command	Change IP configuration	String	<b>mChangeIPCfg</b>
IP	IP-Address	4x UInt_8	<b>0...255 (4x)</b>
Mask	Subnetmask	4x UInt_8	<b>0...255 (4x)</b>
Gateway	Gateway	4x UInt_8	<b>0...255 (4x)</b>

### Response

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

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

### 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.47) 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	<b>sMN</b>
Command	Change IP configuration	String	<b>mChangeEthrCfg</b>
SpeedDuplex	Interface speed and duplex mode	Enum_8	<b>0 = auto (Default)</b> <b>1 = 10MB half</b> <b>2 = 10MB full</b> <b>3 = 100MB half</b> <b>4 = 100MB full</b>

#### Response

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

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

#### 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	<b>sMN</b>
Command	DHCP enable / disable	String	<b>mEnableDHCP</b>
DHCP	Enable/Disable DHCP	Bool_1	<b>0 = disabled (Default)</b> <b>1 = enabled</b>

##### Response

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

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

## 2.4.5 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	<b>sMN</b>
Command		Add landmark	String	<b>mNLAYAddLandmark</b>
LandmarkData		Number of landmarks follow	UInt_16	<b>1...50</b>
Landmark data	X	X-Position	Int_32	<b>-10 000 000 ... +10 000 000 mm</b>
	Y	Y-Position	Int_32	<b>-10 000 000 ... +10 000 000 mm</b>
	Type	Landmark type	Enum_8	<b>0 = undefined 1 = reflector (fix)</b>
	Subtype	Reflector type	Enum_8	<b>0 = undefined 1 = flat 2 = cylindric</b>
	Size	Size: Diameter or width	UInt_16	<b>0...200 mm (Default: 80 mm)</b>
	LayerID	Layer IDs assigned, 1..3 follow	UInt_16	<b>1...3</b>
	<b>Layer</b>	ID	Layers ID	UInt_16

**Response**

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

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

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



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

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

**2.4.7 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	<b>sMN</b>	
Command	Delete landmark	String	<b>mNLAYDeLlandmark</b>	
LandmarkData	Number of landmarks follow for delete	UInt_16	<b>1...50</b>	
<b>LMD</b>	GlobalID	Global ID of the landmark to be deleted	UInt_16	<b>0...11 999</b>

**Response**Command Syntax: **sAN mNLAYDeLlandmark** *errorCode*

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

## 2.4.8 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	<b>sMN</b>
Command		Read landmark	String	<b>mNLAYGetLandmark</b>
LandmarkData		Set the number of landmarks for read	UInt_16	<b>1...50</b>
<b>LMD</b>	GlobalID	Global ID of the transmitted landmark	UInt_16	<b>0...11 999</b>

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

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

### 2.4.9 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	<b>sMN</b>
Command	Get Layer	String	<b>mNLAYGetLayer</b>
LayerID	ID of the Layer	UInt_16	<b>0...319</b>

#### Response

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

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

### 2.4.10 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	<b>sMN</b>
Command	Read Layout	String	<b>mNLAYGetLayout</b>

#### Response

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

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

### 2.4.11 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	<b>sMN</b>
Command	Erase Layout	String	<b>mNLAYEraseLayout</b>
Erase	Select the Memory for erase	Enum_8	<b>0 = ram</b> <b>1 = ram+flash</b>

#### Response

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

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

### 2.4.12 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	<b>sMN</b>
Command	Store Layout permanent	String	<b>mNLAYStoreLayout</b>

#### Response

Command Syntax: **sAN mNLAYStoreLayout errorCode**

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

## 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.31)*).

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	<b>sMN</b>
Command	Do Mapping	String	<b>mNMAPDoMapping</b>

#### Response (Method Acknowledge)

Command Syntax: **sMA mNMAPDoMapping**

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

**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	<b>sAN</b>		
Command		Do Mapping	String	<b>mNMAPDoMapping</b>		
ErrorCode		Error code	Enum_8	<b>0 = no error (Default)</b> <b>1 = wrong operating mode</b> <b>2 = asynchrony Method terminated</b> <b>5 = timeout</b> <b>6 = method already active</b> <b>7 = general error</b>		
LandmarkData		Landmark data available	UInt_16	<b>0 = no data</b> <b>1 = landmark data follow</b>		
<b>LMD</b>	Landmark-Filter		Enum_8	<b>0 = used</b> <b>1 = seen landmark (fix)</b> <b>2 = expected</b>		
	Reflectors		UInt_16	<b>0...40</b>		
	<b>Landmark Data</b>	Cart		UInt_16	<b>0 = no Cartesian data follow</b> <b>1 = Cartesian data follow</b>	
		<b>Cart</b>	X	X-Coordinate	Int_32	<b>-70 000...</b> <b>+70 000 mm</b>
			Y	Y-Coordinate	Int_32	<b>-70 000...</b> <b>+70 000 mm</b>
		Polar		UInt_16	<b>0 = no polar data follow (fix)</b> <b>1 = polar data follow</b>	
		<b>Polar</b>	Dist	Distance	UInt_32	
			Phi	Orientation	UInt_32	
		OptReflectorData		Optional landmark follow	UInt_16	<b>1 = optional data follow (fix)</b>
		<b>Optional</b>	LocalID	Local ID	UInt_16	<b>0...40</b>



			Global ID	UInt_16	<b>0...11 999</b>
			Type	Enum_8	<b>0 = undefined</b> <b>1 = reflector</b>
			Sub-type	Enum_16	<b>0 = undefined</b> <b>1 = flat</b> <b>2 = cylindrical</b>
			Quality	UInt_16	<b>0 = Reserved</b>
			Time-stamp	UInt_32	<b>0...4 294 967 295 ms</b> <b>(0xffffffff)</b>
			Size	UInt_16	<b>1...150 mm</b> <b>(Default: 80mm)</b>
			Hit-Count	UInt_16	<b>1...1440</b>
			Mean-Echo	UInt_16	<b>0...1023</b>
			Index-Begin	UInt_16	<b>0...1439</b>
			Index-End	UInt_16	<b>0...1439</b>

## 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.26). 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	<b>sMN</b>
Command	Get detected landmarks	String	<b>mNLMDGetData</b>
Wait	Set last detected landmark (0) or new landmarks (1)	Bool_1	<b>0 = last detected landmark</b> <b>1 = wait for new landmarks</b>
Mask	Select data format	Enum_8	<b>0 = reflectors</b> <b>1 = reflectors +scan</b>

#### 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	<b>sMA</b>
Command	Asynchronous request of new landmarks	String	<b>mNLMDGetData</b>

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

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

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

## 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.73) 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.25).

#### 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	<b>sMN</b>
Command	Position Request	String	<b>mNPOSGetPose</b>
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	<b>0 = instantly last pose result</b> <b>1 = wait for next pose result</b>

#### 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	<b>sAN</b>
Command		GetPose	String	<b>mNPOSGetPose</b>
Version		Version of POSE Data	UInt_16	<b>0000 ... FFFF</b>
ErrorCode		Error code	Enum_8	<b>0 = no error</b> <b>1 = wrong operating mode</b> <b>2 = asynchrony Method terminated</b> <b>3 = invalid data</b> <b>4 = no position available</b> <b>5 = timeout</b> <b>6 = method already active</b> <b>7 = general error</b>
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	<b>0 = synchronous: last position result</b> <b>1 = asynchronous: new calculated position</b>
PoseData		Set to 1, if pose data follow	UInt_16	<b>0 = no pose data</b> <b>1 = pose data follow</b>
<b>Pose Data</b>	X	X-Position	Int_32	<b>-10 000 000 ...</b> <b>+10 000 000 mm</b>
	Y	Y-Position	Int_32	<b>-10 000 000 ...</b> <b>+10 000 000 mm</b>
	Phi	Orientation	UInt_32	<b>0...360000 mgrad</b>
	OptPoseData	Set to 1, if optional pose data follow	UInt_16	<b>0 = no optional data</b> <b>1 = optional data follow</b>
	OPD	output-Mode	Data are instantly or extrapolated	Enum_8

	Time-stamp	Timestamp	UInt_32	<b>0...4 294 967 295 ms (0xffffffff)</b>
	Mean-Dev	Mean deviation	Int_32	<b>0... 2000 mm</b>
	Nav-Mode	Position mode	Enum_8	<b>0 = initial positioning 1 = continuous positioning 2 = virtual positioning 3 = positioning stopped 4 = position invalid 5 = external</b>
	InfoState	Diagnosis Information	UInt_32	<b>Information is shown in the table below</b>
	NumUsedReflectors	Number of used Reflectors	UInt_8	<b>0...40</b>



**InfoState**

<b>Bit</b>	<b>Description</b>
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.69) 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	<b>sMN</b>
Command	Position data request	String	<b>mNPOSGetData</b>
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	<b>0 = instantly last pose result</b> <b>1 = wait for next pose result</b>
Mask	Select the set of data	Enum_8	<b>0 = pose+reflectors</b> <b>1 = pose+scan</b> <b>2 = pose+reflectors+scan</b>

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

Command Syntax: **sMA mNPOSGetData**

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

### Response after Executing Position Data Request

Command Syntax: **sAN mNPOSGetData** *version errorCode wait poseData [x y phi optPoseData [output-Mode 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	<b>sAN</b>
Command		Position data request	String	<b>mNPOSGetData</b>
Version		Version of positioning data	UInt_16	<b>0000 ... FFFF</b>
ErrorCode		ErrorCode	Enum_8	<b>0 = no error</b> <b>1 = wrong operating mode</b> <b>2 = asynchrony Method terminated</b> <b>3 = invalid data</b> <b>4 = no position available</b> <b>5 = timeout</b> <b>6 = method already active</b> <b>7 = general error</b>
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	<b>0 = synchronous: last position result</b> <b>1 = asynchronous: new calculated position</b>
Mask		Selected set of data	Enum_8	<b>0 = pose+reflectors</b> <b>1 = pose+scan</b> <b>2 = pose+reflectors+scan</b>
PoseData		Pose data follow	UInt_16	<b>0 = no pose data</b> <b>1 = pose data follow</b>
<b>Pose Data</b>	X	X-Position	Int_32	<b>-10 000 000...</b> <b>+10 000 000 mm</b>

	Y	Y-Position	Int_32	<b>-10 000 000... +10 000 000 mm</b>	
	Phi	Orientation	UInt_32	<b>0...360 000 mdeg</b>	
	OptPoseData	Indicates if optional Pose data follow. Selected by showOptParam in NPOSPoseData-Format	UInt_16	<b>0 = no optional data 1 = optional data follow</b>	
<b>Optional Pose Data</b>	Output-Mode	Data are instantly or extrapolated	Enum_8	<b>0 = instantly 1 = extrapolated</b>	
	Time-stamp	Timestamp	UInt_32	<b>0...4 294 967 295 ms (0xffffffff)</b>	
	Mean-Dev	Mean deviation	Int_32	<b>0... 2000 mm</b>	
	Nav-Mode	Position mode	Enum_8	<b>0 = initial positioning 1 = continuous positioning 2 = virtual positioning 3 = positioning stopped 4 = position invalid 5 = external</b>	
	Info-State	Diagnosis Information	UInt_32	<b>reserved</b>	
	QuantUsedReflectors	Number of used Reflectors	UInt_8	<b>0...40</b>	
	LandmarkData	Indicates if landmark data follow	UInt_16	<b>0 = no landmark data 1 = landmark data follow</b>	
<b>Landmark Data</b>	Landmark-Filter	Indicates the type of landmark data ; used, seen or expected	Enum_8	<b>0 = used 1 = seen 2 = expected landmarks</b>	
	Reflectors	Number of landmark Data follow	UInt_16	<b>0...40</b>	
	Reflector Data	Cart	Set to 1 if Cartesian Data follow. Selectable by format in NLMDLandmarkDataFormat	UInt_16	<b>0 = no Cartesian data 1 = Cartesian data follow</b>
	Cart.	X	X-Coordinate	Int_32	<b>-70 000... +70 000 mm</b>

		Y	Y-Coordinate	int_32	<b>-70 000... +70 000 mm</b>
		Polar	Set to 1 if polar Data follow. Selectable by format in NLMDLandmarkDataFormat	UInt_16	<b>0 = no polar data 1 = polar data follow</b>
	<b>Polar</b>	Dist	Distance	UInt_32	<b>0... 70 000 mm</b>
		Phi	Orientation	UInt_32	<b>0...360 000 mdeg</b>
		optLandmarkData	Indicates if optional landmark data follow. Selected by showOptParam in NLMDLandmarkDataFormat	UInt_16	<b>0 = no optional data 1 = optional landmark data follow</b>
	<b>Optional Reflector Data</b>	LocalID	Local ID	UInt_16	<b>0...40</b>
		GlobalID	Global ID	UInt_16	<b>0...11999</b>
		Type	Landmark type	Enum_8	<b>0 = undefined 1 = reflector</b>
		Sub-type	Reflector type	Enum_16	<b>0 = undefined 1 = flat 2 = cylindric</b>
		Quality	Reserved	UInt_16	<b>reserved</b>
		Time-stamp	Timestamp	UInt_32	<b>0...4 294 967 295 ms (0xffffffff)</b>
		Size	Size	UInt_16	<b>1...150 mm (Default: 80mm)</b>
		Hit-Count	Number of hits	UInt_16	<b>1...1440</b>
		Mean-Echo	Mean Echo Amplitude	UInt_16	<b>0...1023</b>
		Index-Begin	Start Index of the reflector	UInt_16	<b>0...1439</b>
		Index-End	End Index of the reflector	UInt_16	<b>0...1439</b>
		NAVscanData	Indicates the number of output channels of the Scan data; Channel with 32 bit values	UInt_16	<b>0 = no scan data 1 = one output channel for scan data (distance) 2 = two Output channels for scan data (distance + echo)</b>

<b>NAVScan-Data</b>	ContentType		Content of the channel.	String	<b>DIST1 : distance ANGL1 : angle</b>
	ScaleFactor		Multiplier	Real	<b>1 (for NAV350 always)</b>
	ScaleOffset		Offset	Real	<b>0 (for NAV350 always)</b>
	StartAngle		Start angle of the Scans	Int_32	<b>0...+360 000 mdeg</b>
	AngleRes		Angular step	UInt_16	<b>1/1000 degree ; 250 mdeg (fix)</b>
	TimestampStart		Timestamp Start of scan	UInt_32	<b>ms</b>
	Data		Number of scan data follow	UInt_16	<b>0...1440</b>
	<b>Scan</b>	AData	Scan data 1 to n. DIST in mm, ANGL in 1/10.000 degree	UInt_32	<b>00000000 ... FFFFFFFF</b>
NAVRemissionData			Indicates a 16 bit channel is enabled for remission data	UInt_16	<b>0 = no remission data 1 = one channel for remission data</b>
<b>Remissions-Scan Data</b>	ContentType		Type of data	String	<b>RSSI1 Remission</b>
	ScaleFactor		Multiplier	Real	<b>1 (for NAV350 always)</b>
	ScaleOffset		Offset	Real	<b>0 (for NAV350 always)</b>
	StartAngle		Start angle of the Scans	Int_32	<b>0...+360 000 mdeg</b>
	AngleRes		Angular step	UInt_16	<b>1/1000 degree ; 250 mdeg (fix)</b>
	TimestampStart		Timestamp at start of Scan	UInt_32	<b>0...4 294 967 295 ms (0xffffffff)</b>
	Data		Number of Scan data	UInt_16	<b>0...1 440</b>
	<b>Scan</b>	AData	Scan data 1 to n	UInt_16	<b>0000...FFFF</b>

### 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	<b>sMN</b>
Command	Set Velocity of the NAV350	String	<b>mNPOSSetSpeed</b>
X	X-component of velocity in the coordinate system defined by <i>coordBase</i>	Int_16	<b>-32 000 ... +32 000 mm/s</b>
Y	Y- component of velocity in the coordinate system defined by <i>coordBase</i>	Int_16	<b>-32 000 ... +32 000 mm/s</b>
Phi	Angular velocity of the NAV350	Int_32	<b>-360 000 ... +360 000 mdeg/s</b>
Timestamp	Timestamp of the Velocity vector related to the NAV350 clock	UInt_32	<b>0...4.294.967.295 ms (0xffffffff)</b>
CoordBase	Coordinate system of the velocity vector, local or global	Enum_8	<b>0 = local coordinate system of the NAV350 1 = absolute coordinate system</b>

#### Response

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

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

### 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	<b>sMN</b>
Command	Set current NAV350 position	String	<b>mNPOSSetPose</b>
X	X-Position of the NAV350	Int_32	<b>-10 000 000... +10 000 000 mm</b>
Y	Y-Position of the NAV350	Int_32	<b>-10 000 000... +10 000 000 mm</b>
Phi	Orientation of the NAV350	Int_32	<b>-360 000... +360 000 mdeg</b>

#### Response

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

Command Syntax: <b>sAN mNPOSSet- Pose</b> <i>errorCode</i> Command part	Description	Type	Range/Value
Command type	Response (SOPAS method by name)	String	<b>sAN</b>
Command	Set current NAV350 position	String	<b>mNPOSSetPose</b>
ErrorCode	Error code	Enum_8	<b>0 = no error 1 = wrong operating mode 3 = parameter out of range 7 = general error</b>



### 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	<b>sMN</b>
Command	Set current NAV350 position	String	<b>mNPOSSetPoseID</b>
LandmarkID	Landmark ID in view of theNAV350	UInt_16	<b>0...11 999</b>

#### Response

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

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

**Australia**

Phone +61 3 9457 0600  
1800 334 802 – tollfree  
E-Mail sales@sick.com.au

**Belgium/Luxembourg**

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

**Brasil**

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

**Canada**

Phone +1 905 771 14 44  
E-Mail information@sick.com

**Česká republika**

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

**China**

Phone +86 4000 121 000  
E-Mail info.china@sick.net.cn  
Phone +852-2153 6300  
E-Mail ghk@sick.com.hk

**Danmark**

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

**Deutschland**

Phone +49 211 5301-301  
E-Mail info@sick.de

**España**

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

**France**

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

**Great Britain**

Phone +44 (0)1727 831121  
E-Mail info@sick.co.uk

**India**

Phone +91-22-4033 8333  
E-Mail info@sick-india.com

**Israel**

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

**Italia**

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

**Japan**

Phone +81 (0)3 3358 1341  
E-Mail support@sick.jp

**Magyarország**

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

**Nederland**

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

**Norge**

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

**Österreich**

Phone +43 (0)22 36 62 28 8-0  
E-Mail office@sick.at

**Polska**

Phone +48 22 837 40 50  
E-Mail info@sick.pl

**România**

Phone +40 356 171 120  
E-Mail office@sick.ro

**Russia**

Phone +7-495-775-05-30  
E-Mail info@sick.ru

**Schweiz**

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

**Singapore**

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

**Slovenija**

Phone +386 (0)1-47 69 990  
E-Mail office@sick.si

**South Africa**

Phone +27 11 472 3733  
E-Mail info@sickautomation.co.za

**South Korea**

Phone +82 2 786 6321/4  
E-Mail info@sickkorea.net

**Suomi**

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

**Sverige**

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

**Taiwan**

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

**Türkiye**

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

**United Arab Emirates**

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

**USA/México**

Phone +1(952) 941-6780  
1 (800) 325-7425 – tollfree  
E-Mail info@sickusa.com

More representatives and agencies  
at [www.sick.com](http://www.sick.com)