



ROHDE & SCHWARZ

Test and Measurement
Division

Operating Manual

Average Power Sensor

R&S NRP-Z11

10 MHz to 8 GHz / 200 pW to 200 mW

1138.3004.02

R&S NRP-Z21

10 MHz to 18 GHz / 200 pW to 200 mW

1137.6000.02

Printed in the Federal
Republic of Germany

Tabbed Divider Overview

Data Sheet

Safety Instructions
Certificate of Quality
EU Certificate of Conformity
List of R&S Representatives

Tabbed Divider

1	Chapter 1: Putting into Operation
2	Chapter 2: Virtual Power Meter
3	Chapter 3: Operation
4	Chapter 4: for future extensions
5	Chapter 5: Remote Control – Basics
6	Chapter 6: Remote Control – Commands
7	Chapter 7: for future extensions



Certificate No.: 2002-36

This is to certify that:

Equipment type	Stock No.	Designation
NRP	1143.8500.02	Power Meter
NRP-B1	1146.9008.02	Sensor Check Source
NRP-B2	1146.8801.02	Second Sensor Input
NRP-B5	1146.9608.02	3rd und 4th Sensor
NRP-B6	1146.9908.02	Rear-Panel Sensor
NRP-Z3	1146.7005.02	USB Adapter
NRP-Z4	1146.8001.02	USB Adapter
NRP-Z11	1138.3004.02	Average Power Sensor
NRP-Z21	1137.6000.02	Average Power Sensor

complies with the provisions of the Directive of the Council of the European Union on the approximation of the laws of the Member States

- relating to electrical equipment for use within defined voltage limits
(73/23/EEC revised by 93/68/EEC)
- relating to electromagnetic compatibility
(89/336/EEC revised by 91/263/EEC, 92/31/EEC, 93/68/EEC)

Conformity is proven by compliance with the following standards:

EN61010-1 : 1993 + A2 : 1995
EN55011 : 1998 + A1 : 1999
EN61326 : 1997 + A1 : 1998 + A2 : 2001

For the assessment of electromagnetic compatibility, the limits of radio interference for Class B equipment as well as the immunity to interference for operation in industry have been used as a basis.

Affixing the EC conformity mark as from 2002

ROHDE & SCHWARZ GmbH & Co. KG
Mühldorfstr. 15, D-81671 München

Munich, 2002-06-27

Central Quality Management FS-QZ / Becker

Table of Contents

1	Putting into Operation	1.1
	Unpacking the sensor	1.1
	Connecting the sensor	1.1
	Operation with the R&S NRP basic unit.....	1.2
	Connecting the sensor to the R&S NRP basic unit.....	1.2
	Connecting the sensor to the DUT	1.2
	PC control.....	1.2
	Hardware and software requirements	1.2
	Operation via the Active USB Adapter R&S NRP-Z3	1.4
	Operation via the Passive USB Adapter R&S NRP-Z4.....	1.5
	Connecting the sensor to the DUT	1.5

Figs.

Fig. 1-1 Displaying the total available power of a USB port 1.3
Fig. 1-2 Configuration with Active USB Adapter R&S NRP-Z3 1.4
Fig. 1-3 Changing the primary adapter 1.4
Fig. 1-4 Configuration with Passive USB Adapter R&S NRP-Z4 1.5

1 Putting into Operation



Follow the instructions below precisely to prevent damage to the sensor – particularly when you are putting it into operation for the first time.

Unpacking the sensor

Remove the sensor from its packing and check that nothing is missing. Inspect all items for damage. If you discover any damage, inform the carrier responsible immediately and keep the packing to support any claims for compensation.

It is also best to use the original packing if the sensor is to be shipped or transported at a later date..



The sensor contains components which can be destroyed by electrostatic discharges. To prevent this happening, never touch the inner conductor of the RF connector and never open the sensor.

Connecting the sensor



To prevent EMI, the sensor must never be operated with its enclosure wholly or partially removed. Only use shielded cables that meet the relevant EMC standards.

Never exceed the maximum RF power limit. Even brief overloads can destroy the sensor.

In many cases, the RF connector only requires manual tightening. However, for maximal measurement accuracy, the RF connector must be tightened using a torque wrench with a nominal torque of 1.36 Nm (12" lbs.).

Operation with the R&S NRP basic unit

Connecting the sensor to the R&S NRP basic unit

The sensor can be connected to the R&S NRP basic unit when it is in operation. The interface connector must be inserted, red marking upwards, into one of the R&S NRP basic unit's sensor connectors. When the sensor is connected, it is detected by the R&S NRP basic unit and initialized.

Connecting the sensor to the DUT

The Sensor R&S NRP-Z11/-Z21 has a male N connector and so can be connected to any standard female N connector. Using light pressure, and keeping the male N connector perpendicular, insert it into the female N connector and tighten the N connector locking nut (right-hand thread).

PC control

Hardware and software requirements

The following requirements must be met if the sensor is to be controlled by a PC via an interface adapter:

- The PC must have a USB port.
- The PC's operating system must support the USB port. This is the case with Windows™ 98, Windows™ ME, Windows™ 2000, Windows™ XP and more recent versions of the Windows™ operating system.
- The USB device drivers in the supplied *NRP Toolkit* software package must be installed.

If these requirements are met, the sensor can be controlled using a suitable application program such as the NrpFlashup program contained in the NRP Toolkit (includes the modules Power Viewer, USB Terminal, Firmware Update and Update S-Parameters).

When you insert the CD-ROM supplied with the NRP, the NRP Toolkit is automatically installed on your PC. The rest of the procedure is self-explanatory.

The sensor can be powered in two ways:

- *Self-powered* from a separate power supply via the Active USB Adapter R&S NRP-Z3.
- *Bus-powered* from the PC or a USB hub with its own power supply (*self-powered hub*) via the Active USB Adapter R&S NRP-Z3 or via the Passive USB Adapter R&S NRP-Z4.

As the sensor is a *high-power device*, there is no guarantee that it can be powered from all types of laptop or notebook in the *bus-powered* mode. To be sure, you should determine the current at the USB connectors beforehand:



- In the Windows™ start menu, select **Settings – Control Panel**
- Select the **System** icon
- Select the **Hardware** tab
- By clicking on the button with that name, start the **Device Manager**
- Open **USB Controller** (all USB controllers, hubs and USB devices are listed here)
- Double-click on **USB Root Hub** or select **Properties** in the context menu (use the right-hand mouse button)
- Select the **Power** tab (Fig. 1-1). If the hub is self-powered and the total power available is, as indicated by **Hub Information**, 500 mA per port, high-power devices can be connected.

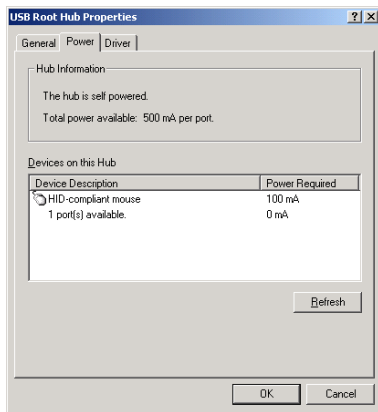


Fig. 1-1 Displaying the total available power of a USB port

If you have any doubts, ask the manufacturer if the USB port on your laptop or notebook can handle *high-power devices*.

Operation via the Active USB Adapter R&S NRP-Z3

Figure 1-2 shows the configuration with the Active USB Adapter R&S NRP-Z3, which also makes it possible to feed in a trigger signal for the *Timeslot* and *Scope* modes. The order in which the cables are connected is not critical.

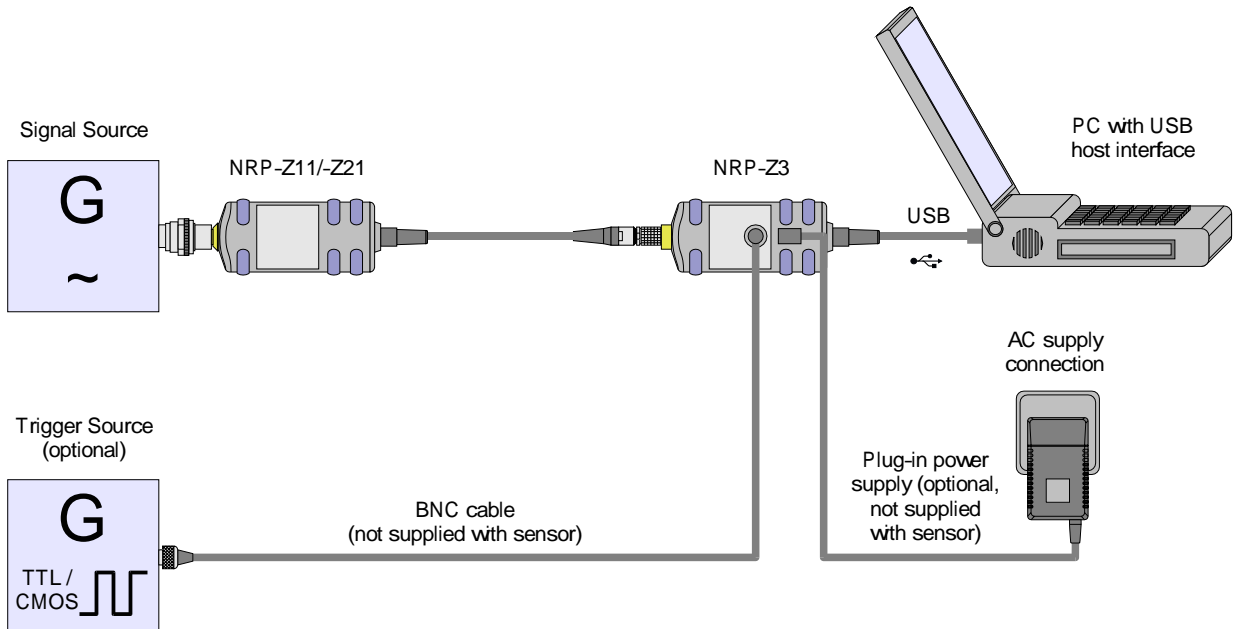


Fig. 1-2 Configuration with Active USB Adapter R&S NRP-Z3

The plug-in power supply for the R&S NRP-Z3 can be powered from a single-phase AC source with a nominal voltage range of 100 V to 240 V and a nominal frequency between 50 Hz and 60 Hz. The plug-in power supply autosets to the applied AC voltage. No manual voltage selection is required.

The plug-in power supply comes with four primary adapters for Europe, the UK, the USA and Australia. No tools of any kind are required to change the primary adapter. The adapter is pulled out manually and another adapter inserted until it locks (Fig. 1-3).

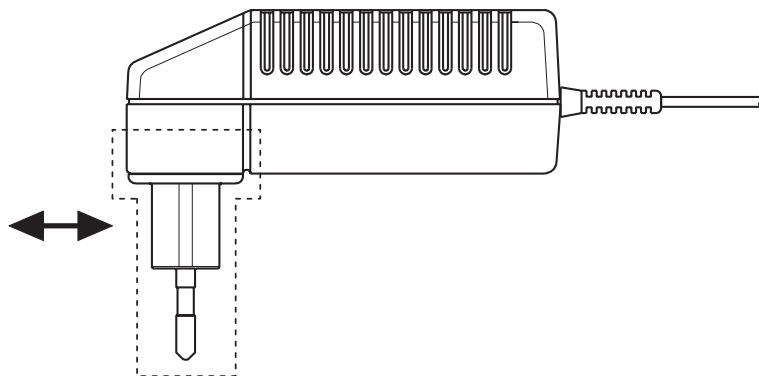


Fig. 1-3 Changing the primary adapter

The plug-in power supply is short-circuit-proof and has an internal fuse. It is not possible to replace this fuse or open the plug-in power supply.



The plug-in power supply is not intended for outdoor use.

Keep within the temperature range of 0°C to 50°C.

If there is any condensation on the plug-in power supply, dry it off before connecting it to the AC supply.

Operation via the Passive USB Adapter R&S NRP-Z4

Fig. 1-4 is a schematic of the measurement setup. The order in which the cables are connected is not critical.

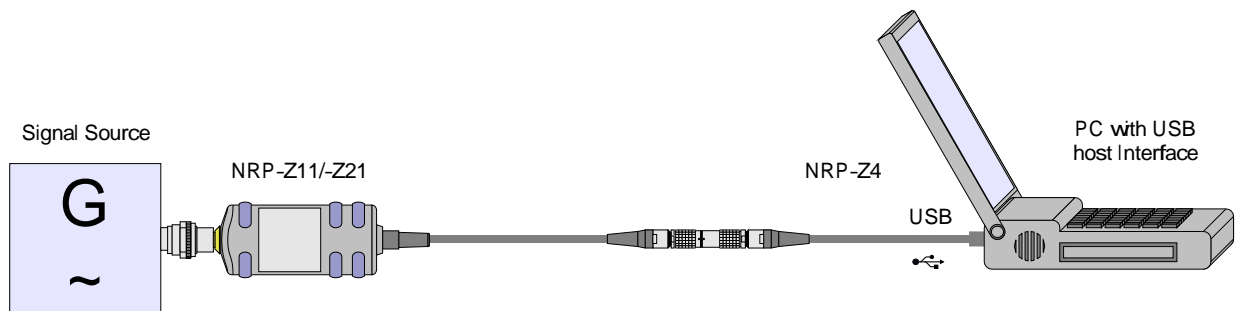


Fig. 1-4 Configuration with Passive USB Adapter R&S NRP-Z4

Connecting the sensor to the DUT

See the section “Operation with the R&S NRP basic unit” for information on how to connect the sensor to the DUT.

Table of Contents

2 Virtual Power Meter 2.1

Overview 2.1

 Menus..... 2.3

Figs.

Fig. 2-1 **Power Viewer** virtual power meter 2.1

Tables

Table 2-1 Virtual power meter keys 2.2
Table 2-2 Virtual power meter entry fields 2.2

2 Virtual Power Meter

You will find the **NrpFlashup** program for controlling sensors with a PC under Windows™ on the CD-ROM that accompanies the sensor. The program comprises several modules which can be started centrally via the Windows™ start-menu entry **NRP Toolkit**.

This section describes the **Power Viewer** program module. This is a virtual power meter which only uses a cut-down set of the sensor's functions. This means that after an extremely brief familiarization period, the user can measure the average power of modulated signals.

The other modules in **NrpFlashup** are described in Chapter 3 of the operating manual (**Terminal** and **Update S-Parameters** modules) or in the service manual (**Firmware Update** module).

Overview

Start the virtual power meter using the **NRP Toolkit – Power Viewer** start-menu entry. The **Power Viewer** program window is displayed (Fig. 2-1).

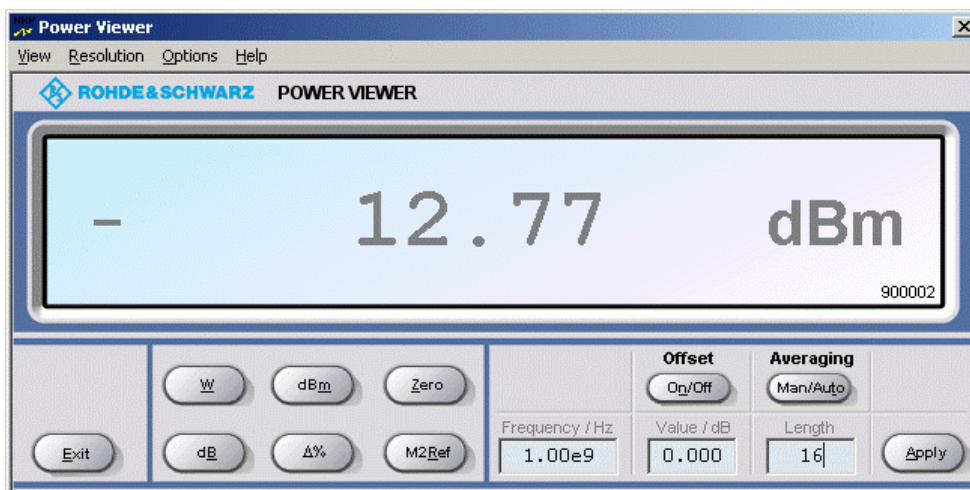


Fig. 2-1 **Power Viewer** – virtual power meter

The result display occupies most of the program window. The result, unit and additional sensor status information are displayed. The serial number of the sensor is displayed in the bottom right. The program window also contains animated buttons and entry fields (see Table 2-1 and Table 2-2).

Table 2-1 Virtual power meter keys

Button	Function	Key combination
Exit	Terminates the program. The current settings are saved and recalled the next time the program is started.	Alt + E
W	Selects Watt as the display unit.	Alt + W
dBm	Selects dBm as the display unit.	Alt + M
Zero	Zeroes the sensor.	Alt + Z
dB	Selects dB as the display unit. This is the log of the ratio of the measured value to the reference value.	Alt + B
Δ%	Selects % as the display unit. The difference between the measured value and the reference value is expressed as a percentage.	Alt + %
M2Ref	Makes the current measured value the reference value for the relative display units dB and %.	Alt + R
Offset On/Off	Turns the offset correction for the sensor on or off. If the offset correction is Off, the Offset/dB entry field has a grey background.	Alt + N
Averaging Man/Auto	Turns auto-averaging on or off. When auto-averaging is on, the Length entry field has a grey background; the current averaging factor is displayed.	Alt + T
Apply	Accepts edited numerical values in the Frequency/Hz , Value/dB and Length entry fields and transfers them to the sensor.	Alt + A or Enter key

Table 2-2 Virtual power meter entry fields

Entry field	Function
Frequency/Hz	Frequency of the RF carrier in Hertz.
Value/dB	Attenuation in dB of the twoport connected to the sensor. The valid range is -100 to 100. The offset correction must be activated beforehand with the Offset On/Off button if this entry field is to be edited.
Length	Length of the averaging filter (= averaging factor). The valid range is 1 to 65536. Averaging must be set to manual with the Averaging Man/Auto button if this entry field is to be edited.

Scientific notation can also be used for the entry fields. If an invalid entry is made, an error message is output. An edited numerical value will not be transferred to the sensor unless you use the **Apply** button or the Enter key to terminate the entry.

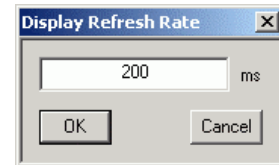
Menus

The menu bar can be used to call less frequently used functions.

View

Display Refresh Rate

Opens a dialog box to adjust the display refresh rate. The time in milliseconds between two refresh operations is entered. The default setting is 200 ms.



Colours

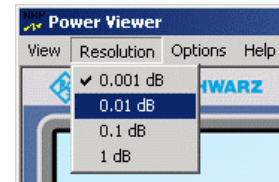
- Result
- Unit
- Edit
- Button

Opens a dialog box to select the background colour for

- the result,
- the unit,
- the text in the number fields or
- the key labelling.

Resolution

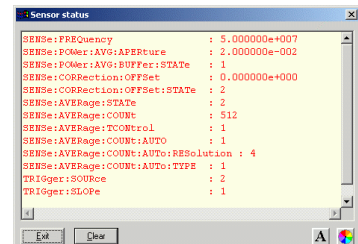
For setting the result resolution. If auto-averaging has been selected, a higher resolution leads to a greater averaging factor, which means a longer result settling time.



Options

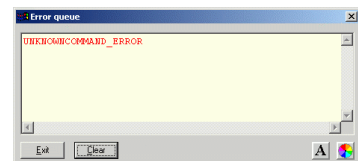
Read Sensor Status ...

Reads the current sensor status. A parameter list is output.



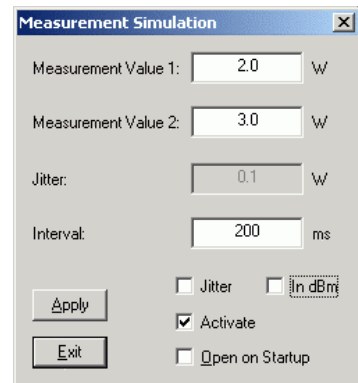
Read Error Queue ...

Reads the error queue. All the error messages that have been issued since the last call are read line-by-line. A tick before this menu entry indicates that an error has occurred.



Simulation ...

For trying out the functions of the virtual power meter without actually connecting a sensor. The display alternates between **Measurement Value 1 & Measurement Value 2** with a period given by **Interval**. Simulation can be activated immediately with the **Activate** check box.



The screenshot shows a dialog box titled "Measurement Simulation" with a close button (X) in the top right corner. It contains the following fields and controls:

- Measurement Value 1: 2.0 W
- Measurement Value 2: 3.0 W
- Jitter: 0.1 W
- Interval: 200 ms
- Buttons: Apply, Exit
- Checkboxes: Jitter, In dBm, Activate, Open on Startup

Reset Sensor

Initializes the sensor. Any previous zeroing remains valid.

Help**Contents**

Opens the table of contents for the online-help facility.

About

Displays information about the program version used, etc.

Table of Contents

- 3 Manual Operation..... 3.1**
 - Program module "Terminal" 3.1**
 - Main control elements 3.1
 - Menus..... 3.3
 - Program module "Firmware Update" 3.6**
 - Program module "Update S-Parameters" 3.6**
 - Fundamentals 3.6
 - Procedure..... 3.9

Figs.

Fig. 3-1 Sending commands using the **Input** field..... 3.1
Fig. 3-2 Sending commands using command files 3.2
Fig. 3-3 Dialog window for loading an s-parameter table 3.9
Fig. 3-4 Dialog window for loading the backup file of a calibration data set 3.10

Tables

Table 3-1 Buttons assigned to the **Input** field..... 3.2
Table 3-2 Buttons assigned to the **Command File** field..... 3.2
Table 3-3 Buttons assigned to the **Output** field..... 3.3
Table 3-4 Uncertainties of the s-parameter test system (example)..... 3.7
Table 3-5 Interpolated uncertainties of measurement frequencies for s-parameters (example)..... 3.7

3 Manual Operation

The previous section describes the Power Viewer program module supplied with the instrument. This module simplifies the most frequently used function of a power meter – measuring the average power of an RF signal of almost any modulation. Other program modules are also part of the supplied equipment and can be selected in the Start menu:

- **Power Viewer:** A detailed description of this virtual power meter module is provided in section 2.
- **Terminal:** Program module for sending commands and command sequences to the sensor and for displaying measurement results, status information and other data from the sensor
- **Firmware Update:** Program module for updating the sensor firmware
- **Update S-Parameters:** Program module for loading an s-parameter table into the sensor

Program module "Terminal"

Main control elements

With the USB terminal, commands and command sequences can be sent to the sensor in two different ways:

- Commands are entered in the **Input** field (Fig. 3-1). Consecutive commands can be entered as separate lines, one below the other. The buttons associated with the **Input** field are described in Table 3-1.
- Commands or command sequences are stored in *command files*. Command files are created with a text editor, for instance, and then stored. They can be called as often as required (Fig. 3-2). The buttons of the **Command File** field are described in Table 3-2.

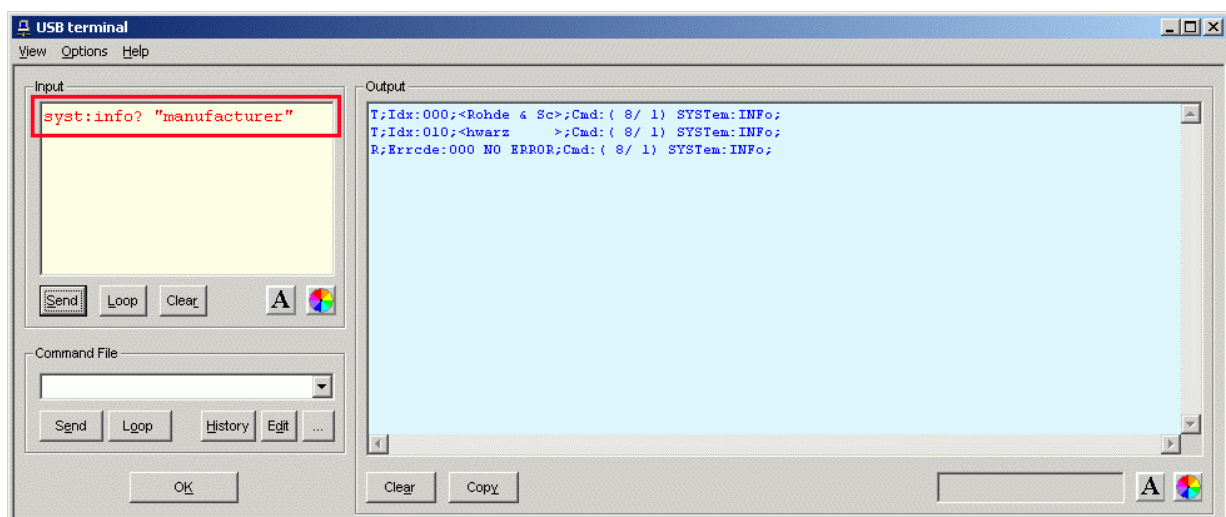


Fig. 3-1 Sending commands using the **Input** field

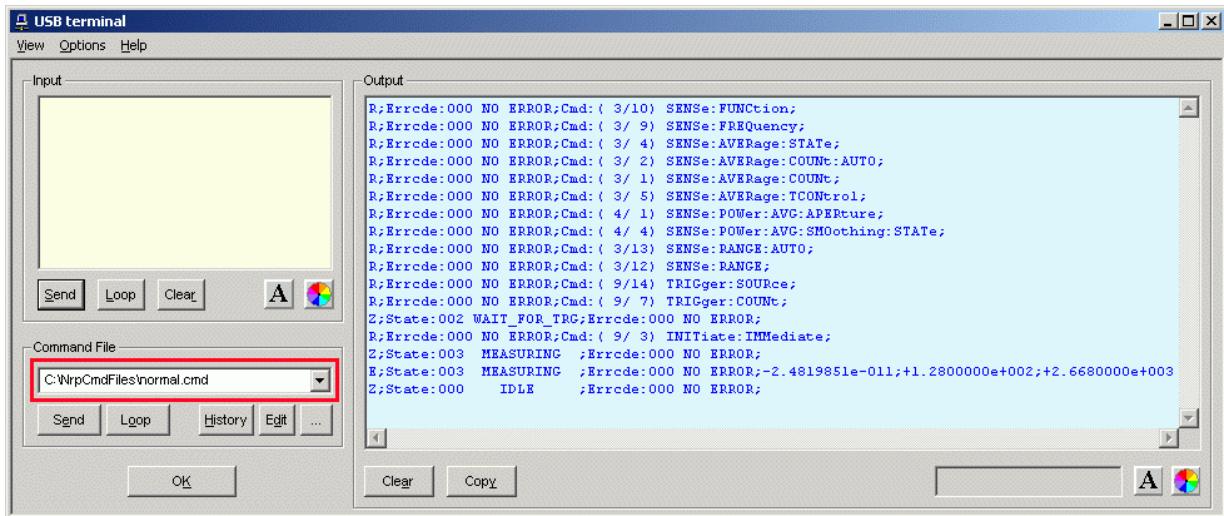


Fig. 3-2 Sending commands using command files

Table 3-1 Buttons assigned to the **Input** field

Button	Function	Key combination
Send	Sends the content of the Input entry field to the sensor.	Alt + S
Loop	With Loop the command or command sequence is cyclically sent. Pressing the button again terminates the cyclic transmission. The repetition rate is set in a dialog window that can be opened with View - Loop...	Alt + L
Clear	Clears the content of the Input field.	Alt + R
Font key	Opens a dialog window where the font for the Input field can be selected.	
Colour key	Opens a dialog window where the background colour of the Input field can be selected.	

Table 3-2 Buttons assigned to the **Command File** field

Button	Function	Key combination
Send	Sends the content of the command file to the sensor.	Alt + E
Loop	With Loop the command or command sequence is cyclically sent. Pressing the button again terminates the cyclic transmission. The repetition rate is set in a dialog window that can be opened with View - Loop...	Alt + O
History	Opens a window for editing the command file name in the Command File field.	Alt + H
Edit	Opens the selected command file in the Windows™ text editor.	Alt + D
...	Opens a file opening dialog for selecting the command file.	

A command line starting with a tab, a blank or a special character is considered a comment and not forwarded to the sensor.

Measurement results, parameters and status information returned by the sensor are displayed in the **Output** field.

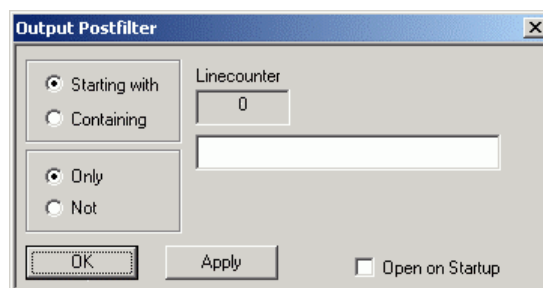
Table 3-3 Buttons assigned to the **Output** field

Button	Function	Key combination
Clear	Clears the content of the Output field	Alt + A
Copy	Copies the content of the Output field to the clipboard. (Another possibility: mark the desired information in the output window with the mouse cursor, press the right mouse key or Ctrl+C and then copy the selected text to the clipboard using the menu item Copy in the opened context menu.)	Alt + Y
Font button	Opens a dialog window where the font for the Output field can be selected.	
Colour button	Opens a dialog window where the background colour of the Output field can be selected.	

Close the USB terminal with OK.

Menus

View Post Filter ... Opens the **Output Postfilter** dialog window where the lines stored in the input buffer can be filtered according to different criteria.



Filter criteria:

Only + Starting with: Only lines starting with the entered character string are displayed.

Not + Starting with: Only lines not starting with the entered character string are displayed.

Only + Containing: Only lines containing the entered character string are displayed.

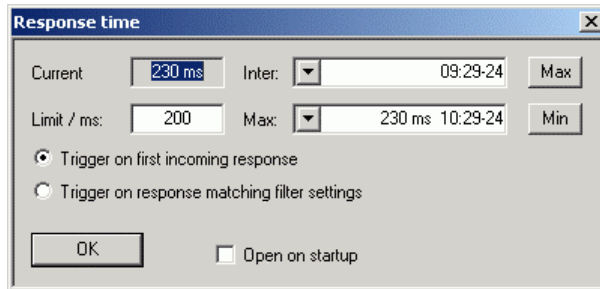
Not + Containing: Only lines not containing the entered character string are displayed.

Lines not matching the specific filter criterion are blanked but not cleared.

Filtering is started with **Apply**. The number of lines matching the filter criterion is displayed in the **Linecounter** field. If **Open on startup** is active, the **Output Postfilter** dialog is automatically opened when the terminal is started. The dialog window is closed with **OK**.

Response Time ...

Opens the **Response time** dialog window where the response time of the sensor can be set.



Current indicates the time elapsed between dispatch of the last command and receipt of an acknowledgement from the sensor. When the **Max** button is clicked, the response times exceeding the value in the **limit / ms** field are recorded. When the **Min** button is clicked, the response times within the value in the **limit / ms** field are recorded.

If **Trigger on first incoming response** is active, the time measurement is terminated as soon as the first response arrives after a command is sent. If **Trigger on response matching filter settings** is active, the time measurement is terminated as soon as the first response matching the filter criterion in the **Output Postfilter** dialog window is received.

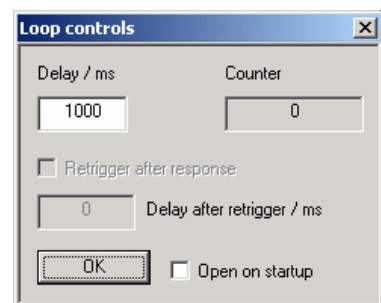
If **Open on startup** is active, the **Response Time** dialog is automatically displayed when the Terminal module is started. The dialog window is closed with **OK**.

Loop ...

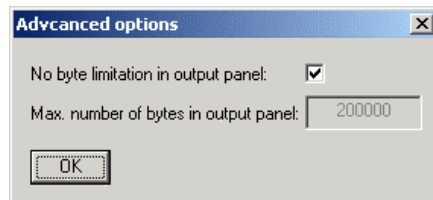
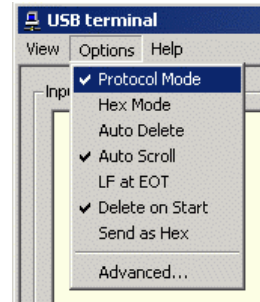
Opens the **Loop controls** dialog window where the cyclic transfer of commands and command sequences can be controlled.

In the **Delay / ms** field, the time interval for the cyclic transfer is specified in milliseconds.

The number of completed transfer cycles is displayed in the **Counter** field. If **Open on startup** is active, the **Response time** dialog is automatically opened when the Terminal module is started. The dialog window is closed with **OK**.



Options	Protocol Mode	In this mode, a time stamp is added to each response block.
	Hex Mode	In this mode, the response blocks from the sensor are displayed in hexadecimal format.
	Auto Delete	With this option active, the Output field is automatically cleared when the Send button is pressed.
	Auto Scroll	With this option active, older items in the Output field are automatically shifted upward and off the display if space is required for new values.
	LF at EOT	With this option active, a line feed is appended to each response block from the sensor.
	Delete on Start	With this option active, the Output field is automatically cleared when the Terminal module is started.
	Send as Hex	With this option active, the text in the Input field is interpreted as a hexadecimal character sequence.
	Advanced ...	Opens a dialog window where the buffer size for the Output field can be set.



Help	Contents	Opens the table of contents for the online help.
	About	Displays information about the program version, etc.

Program module "Firmware Update"

A detailed description of the program module for firmware updates is provided in the Service Manual.

Program module "Update S-Parameters"

Fundamentals

With the Sensor R&S NRP-Z11/Z21 the influence of any twoport connected to the input on the measurement result can be corrected by way of calculation. A precondition is that a complete set of s-parameters of the twoport is available in the frequency range in question. The set of calibration data in the R&S NRP-Z11/-Z21 therefore includes an s-parameter table with up to 1000 measurement frequencies. The real and the imaginary part of each frequency as well as the uncertainty of s-parameters s_{11} , s_{12} , s_{21} and s_{22} can be stored. Since the measurement frequencies in the s-parameter table are independent of the calibration frequencies, they can be set so that the twoport frequency range of interest is optimally covered. The real and the imaginary parts between these measurement frequencies are linearly interpolated, while the more substantial measurement uncertainty at the two neighbouring frequency points is used for calculating the uncertainty of the measurement result. Below the first and above the last measurement frequency, the values of the first and the last measurement frequency are used, respectively.

The NrpFlashup program (menu item **Update S-Parameters**) is used for loading an s-parameter table. To ensure compatibility with a great number of network analyzers, NrpFlashup can process measurement data files in S2P format. All standard frequency units (Hz, kHz, MHz, GHz) and display formats (real and imaginary part, linear magnitude and phase, magnitude in dB and phase) are supported. The only restriction is that a reference impedance of 50 Ω must be used for the s-parameters. Other noise parameters in the measurement data file are not evaluated.

Structure of the S2P measurement data file:

1. The *option line* has the following format:

```
# [<frequency unit>] [<parameter>] [<format>] [<R n>]
```

identifies the *option line*.

The <frequency unit> may be Hz, kHz, MHz or GHz. If a frequency unit is not specified, GHz is implicitly assumed.

If a parameter is specified, S must be used in <parameter> for s-parameter files. If a parameter is not specified, S is implicitly assumed.

The <format> may be MA (linear magnitude and phase in degree), DB (magnitude in dB, phase in degree) or RI (real and imaginary part). If a format is not specified, MA is implicitly assumed.

R is optional and followed by the reference impedance in Ω . If an entry is made for R, R50 must be specified. If no entry is made, R50 is implicitly assumed.

The *option line* should therefore read:

```
# [HZ | KHZ | MHZ | GHZ] [S] [MA | DB | RI] [R 50]
```

2. The measurement frequencies in ascending order are specified as follows:

$$f_i \quad s_{11}(f_i) \quad s_{21}(f_i) \quad s_{12}(f_i) \quad s_{22}(f_i),$$

where $s_{jk}(f_i)$ is the specified display format for the *option line*.

$|s_{jk}(f_i)| \quad \arg s_{jk}(f_i)$ (display format for linear magnitude and phase in degree) or

$20 \cdot \lg |s_{jk}(f_i)| \quad \arg s_{jk}(f_i)$ (display format for magnitude in dB and phase in degree)

$\operatorname{Re} [s_{jk}(f_i)] \quad \operatorname{Im} [s_{jk}(f_i)]$ (display format for real and imaginary part)

3. Comments: Any line starting with an exclamation mark (!) is interpreted as a comment line.

To characterize the measurement uncertainty of the s-parameter test system, another data file can optionally be created. Without this file, the measurement uncertainty cannot be correctly calculated in the sensor. The syntax of the uncertainty data file is similar to that of the S2P data file but U is specified as <Parameter> in the *option line* so that the *option line* reads # Hz U for frequencies in Hz.

The measurement frequencies must not be identical to those of the S2P measurement data files. In most cases a few entries will be sufficient to characterize the measurement uncertainty of the s-parameter test system. An s-parameter uncertainty as high as that of the neighbouring measurement frequencies of the uncertainty data file is then selected. If different values are available, the higher one is chosen. This is illustrated in the example below:

Table 3-4 Uncertainties of the s-parameter test system (example)

f in GHz	unc [$S_{ik}(f)$]
0.1	0.01
1.0	0.01
1.1	0.005
10.0	0.005
10.1	0.01
40.0	0.01

Table 3-5 Interpolated uncertainties of measurement frequencies for s-parameters (example)

f in GHz	unc [$S_{ik}(f)$]
0.9	0.01
0.95	0.01
1.0	0.01
1.05	0.01
1.1	0.005
1.15	0.005
1.2	0.005

At 1.05 GHz, the higher uncertainty of the two adjacent 1.0 GHz and 1.1 GHz measurement frequencies is entered in the s-parameter table. If an uncertainty of 0.005 is desired for all frequencies above 1.0 GHz, the first measurement frequency in the uncertainty data file must be above 1.0 GHz, e.g. 1.000001 GHz.

Structure of the uncertainty data file:

1. The *option line* has the following format:

[<frequency unit>] <parameter> [<format>] [<R n>]

identifies the *option line*.

The <frequency unit> may be Hz, kHz, MHz or GHz. If a frequency unit is not specified, GHz is implicitly assumed.

U must be specified for <parameter> in uncertainty data files. If a parameter is not specified, S is implicitly assumed and as a result an error message is triggered.

<format> is ignored in uncertainty measurement files; the entry is therefore irrelevant.

R is optional and followed by the reference impedance in Ω . If an entry is made for R, R50 must be specified. If no entry is made, R50 is implicitly assumed.

The *option line* should therefore read:

[HZ | KHZ | MHZ | GHZ] U [MA | DB | RI] [R 50]

2. Measurement frequencies in ascending order are specified in the following form:

f_i unc [$s_{11}(f_i)$] unc [$s_{21}(f_i)$] unc [$s_{12}(f_i)$] unc [$s_{22}(f_i)$].

The s-parameters uncertainties are forwarded as follows:

- as extended absolute uncertainties ($k = 2$) for the magnitude of reflection parameters s_{11} and s_{22} , for instance 0.015,
- as extended uncertainties ($k = 2$) in dB for the magnitude of transmission parameters s_{21} and s_{12} , for instance 0.015.

3. Comments: Any line starting with an exclamation mark (!) is interpreted as a comment line.

Two additional values must be specified when the s-parameters are loaded: the lower and the upper nominal measurement limit of the sensor-twoport combination. If s-parameter correction is active, these values are transferred by the sensor in response to SYSTem:INFO? The values cannot always be derived from the lower or upper measurement limit of the sensor alone and from the loss or gain of the preconnected twoport. The upper measurement limit of the sensor-twoport combination may also be limited by the twoport's maximum power-handling capacity. Furthermore, the lower measurement limit may be raised not only by the loss but also by the inherent noise of the twoport. For this reason, NrpFlashup allows these values to be entered.



The upper nominal measurement limit of the sensor-twoport combination entered when loading the s-parameters should be carefully specified, as automatic test systems may evaluate it and an incorrect value may cause the sensor and/or the twoport to be overloaded.

Procedure

To load an s-parameter table into the calibration set of the sensor, proceed as follows:

1. Connect the sensor to the USB port of the PC and start NrpFlashup.
2. Activate **Update S-Parameters** in the menu. The corresponding dialog window is opened (Fig. 3-3).
3. Under **S-Parameter File** enter the search path and the name of the S2P file containing the parameters. Press the **Browse...** button to open a file-opening dialog where the S2P measurement data file can be easily selected.
4. Under **Uncertainty File** enter the search path and the name of the measurement uncertainty file containing the measurement uncertainty of the s-parameter test system. Press the **Browse...** button to open a file-opening dialog where the measurement uncertainty file can be easily selected.
5. Enter the upper and lower nominal measurement limit of the sensor-twoport combination in the **Lower Power Limit** and **Upper Power Limit** fields.
6. Enter a name for the loaded s-parameter set in the **S-Parameter Device Mnemonic** field. This name can later be queried with `SYSTEM:INFO? "SPD Mnemonic"` and is displayed on the NRP basic unit when s-parameter correction is switched on.
7. Activate **S-Parameter Correction on by Default** if the `SENSe:CORRection:TRANsmission` switch should be automatically set to `ON` when the sensor is put into operation.
8. Press **Start** for loading. (The dialog is closed with **OK** and the set parameters are retained. When the dialog is exited with **Cancel**, all parameter modifications are ignored.)

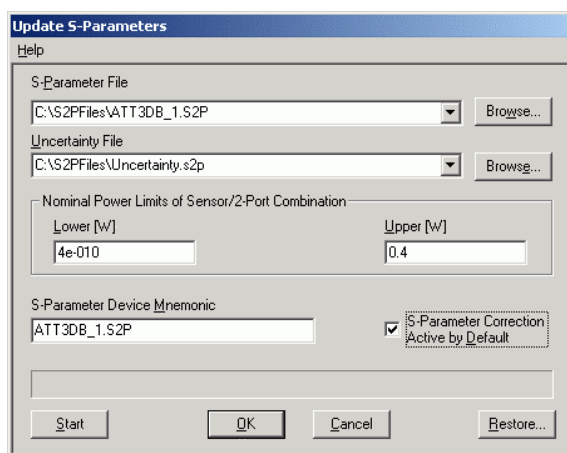


Fig. 3-3 Dialog window for loading an s-parameter table

During loading, the current calibration data set of the sensor is overwritten. To be on the safe side, a backup copy of the current calibration data set is therefore automatically stored before s-parameters are loaded. The names of the backup files have the structure `<batch number>_<date><time>.bak`, where `<batch number>` is the batch number of the sensor, `<date>` the date of the s-parameter update in `yymmdd` format and `<time>` the time of the s-parameter update in the format `hhmmss`.



Store the automatically created backup files on a separate data medium (e.g. diskette, CD-ROM or network drive) and, if required, assign a meaningful name to them to simplify reloading. With the aid of these files, a previously used calibration data set of the sensor can be restored.

To reload the backup file of a calibration data set into the sensor, proceed as follows:

- Press the **Restore...** button. The **Restore S-Parameters** window is opened (Fig. 3-4).
- Enter the search path and the name of the backup file in the **Backup File** field. Press the **Browse...** button to open a dialog where the backup file can be easily selected.
- Press **OK** to start the restore procedure. (With **Cancel** the dialog window is exited without data being restored).

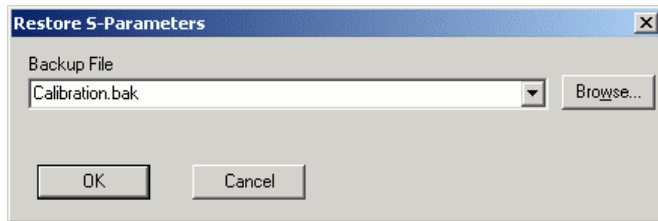


Fig. 3-4 Dialog window for loading the backup file of a calibration data set

Table of Contents

5	Remote Control – Fundamentals	5.1
	USB Driver Stack	5.1
	Universal USB-Driver Interface	5.2
	Functions for Setting the Sensor Mode.....	5.4
	NrpChannelAssignment	5.4
	NrpStartApplicationMode	5.4
	Functions for Sending Commands and Data	5.5
	NrpSendBinaryBlock	5.5
	NrpSendCommand.....	5.5
	Functions for Fetching Results and Data.....	5.6
	NrpDataAvailable	5.6
	NrpGetData	5.6
	NrpGetBinaryResult	5.11
	NrpGetBitfieldResult.....	5.11
	NrpGetFloatArray	5.12
	NrpGetFloatResult.....	5.12
	NrpGetLongResult.....	5.12
	NrpGetStringResult	5.13
	Functions for Status and Error Detection.....	5.14
	NrpEmptyAllBuffers	5.14
	NrpEmptyErrorQueue.....	5.14
	NrpGetLastError	5.14
	NrpGetErrorText.....	5.16
	NrplsAlive	5.16
	NrpGetTriggerState	5.16
	NrpGetTriggerStateText.....	5.17
	Callback Functions.....	5.17
	Functions for Setting Callback Functions.....	5.19
	NrpSetNotifyCallbackCommandAccepted	5.19
	NrpSetNotifyCallbackDataAvailable	5.19
	NrpSetNotifyCallbackDeviceChanged	5.19
	NrpSetNotifyCallbackErrorOccured	5.19
	NrpSetNotifyCallbackStateChanged	5.19
	Windows Messages	5.20

USB-specific Commands	5.20
NrpOpenDriver	5.20
NrpCloseDriver	5.20
NrpClearDevice	5.20
NrpGetDeviceChangedDevName	5.20
NrpGetDeviceChangedMsgText	5.21
NrpGetDeviceID	5.22
NrpGetIDByLogicalName	5.22
NrpGetLogicalName	5.22
NrpGetManufacturerCode	5.22
NrpGetNrOfDevices	5.23
NrpGetProductID	5.23
NrpGetSerialNumber	5.23
NrpSDRGetDescriptor	5.23
NrpSelectDevice	5.24
Examples of Application	5.25
Single Average Power Measurement	5.25
Quick and Multiple Average Power Measurement	5.26
Measuring a Single Burst	5.26
Measurement of a Frame with Several Timeslots	5.27
Scope Mode for Displaying Periodic Signals	5.27
Scope Mode for Displaying a Single Process (Single-Shot Measurement)	5.28

Figs.

Fig. 5-1 Description of dynamic link library NrpControl.dll..... 5.3

Tables

Table 5-1 Components of the USB driver stack 5.1
Table 5-2 Meaning of data types 5.6
Table 5-3 Meaning of group and parameter numbers..... 5.7
Table 5-4 Possible sensor error states 5.14
Table 5-5 Possible sensor trigger states 5.16
Table 5-6 Description of structure *USB_DEVICECHANGED_HANDLER_RESULT*..... 5.21

5 Remote Control – Fundamentals

USB Driver Stack

A USB driver stack is installed for sensors R&S NRP-Z11/-Z21 at the same time as the NRP toolkit. The driver stack comprises the following files:

Table 5-1 Components of the USB driver stack

File	Function	Remarks
Z11USB.sys Z21USB.sys	USB device driver for R&S NRP sensors	
Z11USB.inf Z21USB.inf	Contains information on the installation of the USB device driver	
NrpFU.sys	USB device driver for R&S NRP sensors	Required for firmware update
NrpFU.inf	Contains information on the installation of the USB device driver	Required for firmware update
NrpControl.h	Contains the C declarations of all function calls	Should be included into the part of a C project in which the DLL is addressed *)
NrpControl.lib	Contains the symbols for the linker exported by the DLL	Should be made known to the linker of the development environment *)
NrpControl.dll	Holds the USB device driver to ensure more convenient communication with the sensor	Should be either in the same directory as the executable application or at least in a directory specified in the system path, e.g. %SYSTEM_ROOT%\system32; (%SYSTEM_ROOT% is an environment variable created by Windows™ that contains the Windows directory.) The file NrpControl.dll is copied to the system directory during the installation.

With the supplied components of the driver stack it is possible to address the sensor under various programming languages without involving much additional effort. In C, C++ and CVI projects, header file *NrpControl.h* and program library *NrpControl.lib* must be integrated.

*) The files NrpControl.h and NrpControl.lib are saved to the directory C:\Program Files\Rohde&Schwarz\NRP-Toolkit\NRPControl during the installation of the NRP toolkit (installation performed with the setup type 'Typical').

Universal USB-Driver Interface

The sensor R&S NRP-Z11/-Z21 is addressed by the host PC via SCPI commands (see section 6 of this user manual). These commands are transmitted as ASCII strings to the sensor via the USB and interpreted by the sensor. The sensor sends measured values, parameters and other data in a binary block format.

The software developer can use the dynamic link library *NrpControl.dll* as a universal interface to remote-control the sensor R&S NRP-Z11/-Z21. This interface has an output command for SCPI commands, a command for sending binary blocks (for transmitting calibration data sets to the sensor) and several commands for querying the data sent by the sensor (measured values, device status, USB-specific information, etc.).

The binary block format used in the reverse direction should be interpreted depending on the type of data. If the sensor sends *FLOAT* values, for example, they should be either measured values or queried setting parameters. If they are setting parameters, the exact ones involved should be defined. The DLL provides mechanisms for this interpretation. Fig. 5-1 shows the basic operation of the DLL.

Within the DLL, the following data is managed in independent dynamic buffers:

- Errors that occurred
- Result descriptors (detailed information on the type of buffered data, e.g. measured values, parameters, limit values)
- Binary data
- *FLOAT* data
- *FLOAT* array data
- *LONG* data
- Bit-field data

These buffers are read out and emptied via special read functions. Functions are also available to delete these buffers. The buffers are organized as FIFO memories, i.e. the oldest data stored in a defined buffer is read and removed from the buffer.

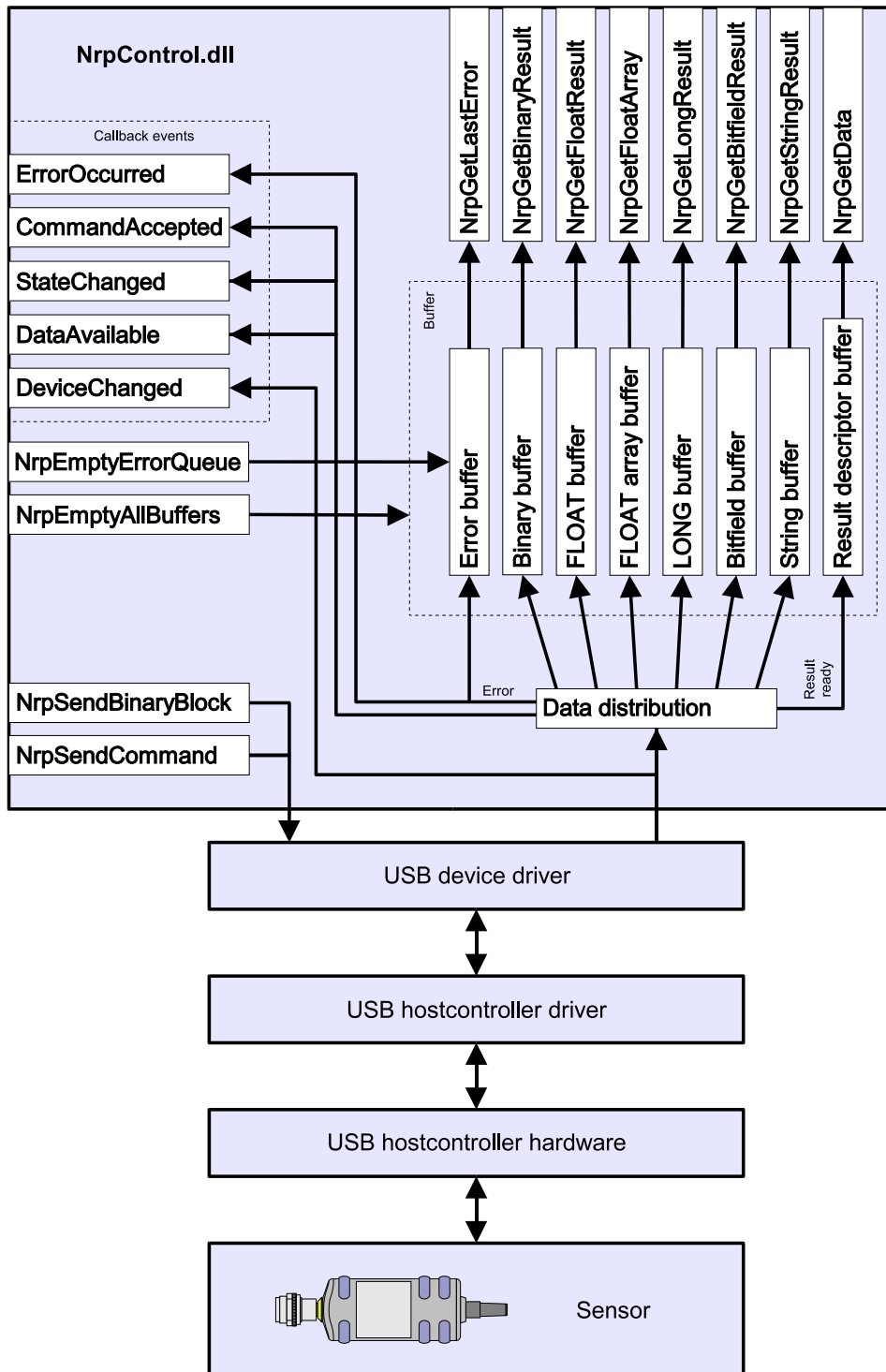


Fig. 5-1 Description of dynamic link library NrpControl.dll

Functions for Setting the Sensor Mode

NrpChannelAssignment

Calling *NrpChannelAssignment* opens a dialog window in which sensors can be assigned a straightforward designation. This is to facilitate operation with several sensors. The sensors are uniquely identified by means of the type designation and the serial number.

Prototype: `void NrpChannelAssignment (void);`

NrpStartApplicationMode

NrpStartApplicationMode is used to set the sensor to the measurement mode.

After a power-on reset the boot loader first starts in the R&S NRP sensors. A firmware update can be performed even if the measurement firmware is defective. After a wait time of 10 s, the sensors automatically switch to the measurement mode. *NrpStartApplicationMode* () immediately changes the mode.

Prototype: `void NrpStartApplicationMode (void);`

Functions for Sending Commands and Data

NrpSendBinaryBlock

NrpSendBinaryBlock is used to send a command, followed by a block of binary data as parameters to the sensor.

Prototype:

```
void NrpSendBinaryBlock (const char *i_pcCommand, void *i_pBuf,  
                        long i_wCount);
```

Parameters:

<i>i_pcCommand</i>	Pointer to the command to be sent (zero-terminated string)
<i>i_pBuf</i>	Pointer to the buffer containing the binary data
<i>i_wCount</i>	Number of bytes to be sent

Example: `NrpSendBinaryBlock ("CALibration:DATA", caldata_ptr, 1234);`

The command *CALibration:DATA* is used to transmit a 1234-byte calibration data set to the flash memory of the sensor. *i_pBuf* points to the beginning of the memory area containing the calibration data.

NrpSendCommand

NrpSendCommand is used to send commands as ASCII strings to the sensor. The function then waits until either the sensor confirms command execution or the wait time exceeds the specified timeout. The function supplies 1 as a return value if the command was successfully executed and 0 if this is not the case.

If the value 0 is transmitted for *i_wTimeoutMs*, the function immediately returns the value 1 without waiting for a confirmation by the sensor.

Section 6 of this user manual provides an overview on the available commands.

Prototype: `long NrpSendCommand (const char *i_pcCommand, long i_wTimeoutMs);`

Parameter: *i_pcCommand* Pointer to the command to be sent (zero-terminated string)

Example: `NrpSendCommand ("*IDN?");`

The query **IDN?* is sent to the sensor.

Functions for Fetching Results and Data

NrpDataAvailable

NrpDataAvailable returns 1 if data is received from the sensor, and 0 if this is not the case and all internal buffers are empty. If a callback function is used to respond to the presence of data, it is not meaningful to use *NrpDataAvailable* at the same time.

Prototype: `long NrpDataAvailable (void);`

NrpGetData

NrpGetData determines the type of the result to be fetched (see Table 5-2) as well as the group number and the parameter number if the result is a parameter. The group number is the consecutive number of a group of associated parameters or commands. Different groups of parameters/commands are implemented for different sensor types depending on their characteristics. The parameter number is the consecutive number of a parameter/command within its group. Group and parameter numbers are 0 for measurement results.

Prototype:

`void NrpGetData (long *o_pDataType, long *o_pGroupNr, long *o_pParamNr);`

Parameters:

- `o_pDataType` Pointer to the *LONG* variable that has to store the ordinal number of the data type
- `o_pGroupNr` Pointer to the *LONG* variable that has to store the group number
- `o_pParamNr` Pointer to the *LONG* variable that has to store the parameter number

Table 5-2 Meaning of data types

Data type (enum constant)	Ordinal number	Data to be fetched
DATA_BINARYBLOCK	0	A binary byte sequence can be fetched with <i>NrpGetBinaryResult</i> (calibration data set of the sensor with command <i>CALibration:DATA?</i>).
DATA_BITFIELDLIMIT	1	Three bit-field values can be fetched with <i>NrpGetBitfieldResult</i> . The first value is the default setting, the second value the bit mask of all permissible states and the third value is not used and assigned with 0.
DATA_BITFIELDPARAM	2	Three bit-field values for displaying discrete states (e.g. <i>OFF</i> , <i>ON</i> , <i>ONCE</i>) can be fetched with <i>NrpGetBitfieldResult</i> . The first value is a parameter value; the second and third values are not used and assigned with 0.
DATA_BITFIELDFEATURE	3	Three bit-field values for displaying implemented commands can be fetched with <i>NrpGetBitfieldResult</i> . The first value contains the information (a bit is set for each command or parameter within a command group, starting with bit 0 for parameter No. 1); the second and third values are not used and assigned with 0.
DATA_FLOATARRAY	4	An array of <i>FLOAT</i> values (power values) and an array of <i>LONG</i> values (indices) can be fetched with <i>NrpGetFloatArray</i> .
DATA_FLOATLIMIT	5	Three <i>FLOAT</i> values (default setting, upper limit value and lower limit value) can be fetched with <i>NrpGetFloatResult</i> .
DATA_FLOATPARAM	6	Three <i>FLOAT</i> values can be fetched with <i>NrpGetFloatResult</i> . The first value is a parameter value; the second and the third value are not used and assigned with 0.0.

Data type (enum constant)	Ordinal number	Data to be fetched
DATA_FLOATRESULT	7	Three <i>FLOAT</i> values (power value and two secondary measured values) can be fetched with <i>NrpGetFloatResult</i> .
DATA_LONGLIMIT	8	Three <i>LONG</i> values (default setting, upper limit value and lower limit value) can be fetched with <i>NrpGetLongResult</i> .
DATA_LONGPARAM	9	Three <i>LONG</i> values can be fetched with <i>NrpGetLongResult</i> . The first value is a parameter value; the second and third values are not used and assigned with 0.
DATA_STRING	10	A string can be fetched with <i>NrpGetStringResult</i> .

Table 5-3 Meaning of group and parameter numbers

Group number	Group of commands	Parameter number	Parameter or command
1	CALibration	1	CALibration:DATA
		2	CALibration:ZERO:AUTO
		3	CALibration:DATA:LENGth
2	INPut (currently reserved)		
3	SENSe	1	SENSe:AVERage:COUNt
		2	SENSe:AVERage:COUNt:AUTO
		3	SENSe:CORRection:DCYClE
		4	SENSe:AVERage:STATe
		5	SENSe:AVERage:TCONtrol
		6	SENSe:CORRection:OFFSet
		7	SENSe:CORRection:OFFSet:STATe
		8	SENSe:DCYClE:OFFSet:STATe
		9	SENSe:FREQuency
		10	SENSe:FUNcTion
		11	SENSe:EUNCertainty:STATe
		12	SENSe:RANGE
		13	SENSe:RANGE:AUTO
		14	SENSe:RANGE:CLEVel

Group number	Group of commands	Parameter number	Parameter or command
		15	SENSe:TIMing:EXCLude:START
		16	SENSe:TIMing:EXCLude:STOP
		17	SENSe:SAMPling
		18	SENSe:AVERage:COUNT:AUTO:MTIME
		19	SENSe:AVERage:COUNT:AUTO:RESolution
		20	SENSe:AVERage:COUNT:AUTO:SLOT
		21	SENSe:AVERage:COUNT:AUTO:NSRatio
		22	SENSe:AVERage:COUNT:AUTO:TYPE
		23	SENSe:CORRection:SPDevice:STATe
		24	SENSe:SGAMma:MAGNitude
		25	SENSe:SGAMma:PHASe
		26	SENSe:SGAMma:CORRection:STATe
		27	SENSe:SGAMma:EUNCertainty
		28	SENSe:EUNCertainty:SGAMma:STATe
4	SENSe:POWer:AVG	1	SENSe:POWer:AVG:APERture
		2	SENSe:POWer:AVG:BUFFer:SIZE
		3	SENSe:POWer:AVG:BUFFer:STATe
		4	SENSe:POWer:AVG:SMOothing:STATe
5	SENSe:POWer:TSLot:AVG	1	SENSe:POWer:TSLot:AVG:COUNT
		2	SENSe:POWer:TSLot:AVG:WIDTh
6	SENSe:POWer:BURSt:AVG	1	SENSe:POWer:BURSt:DTOLerance
7	SENSe:SWEep	1	SENSe:SWEep:AVERage:STATe
		2	SENSe:SWEep:OFFSet:TIME
		3	SENSe:SWEep:POINts
		4	SENSe:SWEep:REALtime
		5	SENSe:SWEep:TIME
		6	SENSe:SWEep:AVERage:COUNT

Group number	Group of commands	Parameter number	Parameter or command
		7	SENSe:SWEEp:AVERage:COUNT:AUTO
		8	SENSe:SWEEp:AVERage:COUNT:AUTO:MTIME
		9	SENSe:SWEEp:AVERage:COUNT:AUTO:RESolution
		10	SENSe:SWEEp:AVERage:COUNT:AUTO:POINT
		11	SENSe:SWEEp:AVERage:COUNT:AUTO:NSRatio
		12	SENSe:SWEEp:AVERage:COUNT:AUTO:TYPE
		13	SENSe:SWEEp:AVERage:TCONtrol
8	SYSTem	1	SYSTem:INFO
		2	SYSTem:INITialize
		3	SYSTem:TRANsaction:BEgin
		4	SYSTem:TRANsaction:END
		5	SYSTem:MINPower
9	Trigger-system commands	1	ABORt
		2	INITiate:CONTinuous
		3	INITiate:IMMediate
		4	reserved
		5	reserved
		6	TRIGger:ATRigger:STATe
		7	TRIGger:COUNt
		8	TRIGger:DELay
		9	TRIGger:DELay:AUTO
		10	TRIGger:HOLDoff
		11	TRIGger:IMMediate
		12	TRIGger:LEVel
		13	TRIGger:SLOPe
		14	TRIGger:SOURce
		15	TRIGger:HYSTeresis

Group number	Group of commands	Parameter number	Parameter or command
10	SERVice	1	SERVice:DITHer
		2	SERVice:SAMPle
		3	SERVice:SIMCount
		4	SERVice:UNLock
		5	SERVice:CALibration:ZERO:NEG0
		6	SERVice:CALibration:ZERO:POS0
		7	SERVice:CALibration:ZERO:NEG1
		8	SERVice:CALibration:ZERO:POS1
		9	SERVice:CALibration:ZERO:NEG2
		10	SERVice:CALibration:ZERO:POS2
		11	SERVice:CALibration:DITHer
		12	SERVice:CALibration:DITHer:DATA
		13	SERVice:CALibration:TEMP
		14	SERVice:CALibration:TEMP:DATA
		15	SERVice:PARAmeter:RTemp
		16	SERVice:PARAmeter:RNULL0
		17	SERVice:PARAmeter:RNULL1
		18	SERVice:PARAmeter:RNULL2
		19	SERVice:PARAmeter:RBAHN
		20	SERVice:PARAmeter:NREF
		21	SERVice:PARAmeter:ATHERM
		22	SERVice:PARAmeter:BTHERM
		23	SERVice:MVCorrection
		24	SERVice:CALibration:TEST
		25	SERVice:RCOunt
		26	SERVice:RESult
		27	SERVice:PARAmeter:CTHERM

Group number	Group of commands	Parameter number	Parameter or command
		28	SERVice:PARAmeter:DThERm
		29	SERVice:PARAmeter:CJUNC
11	IEEE 488.2 Common Commands	1	*RST
		2	*TRG
		3	*IDN?
		4	*TST?
12	TEST	1	TEST:SENSor

NrpGetBinaryResult

NrpGetBinaryResult reads from the binary buffer all response data blocks sent by the sensor since the function was last called and copies them to a buffer. The read bytes are deleted from the binary buffer. The oldest response data blocks are read out first. The function returns the number of bytes that are actually deleted from the binary buffer.

If the user transfers a zero pointer as argument *o_pBuf* to the function, the function returns the number of bytes contained in the binary buffer.

Prototype: `long NrpGetBinaryResult (void *o_pBuf, long i_wCount);`

Parameters: *o_pBuf* Pointer to the buffer containing the result
i_wCount Buffer size in bytes

NrpGetBitfieldResult

Parameters that can accept a specific number of discrete states (e.g. *OFF*, *ON* and *ONCE*) are coded as 32-bit bit fields. Each bit represents one of the possible states. In this way, the sensor can accept or exclude discrete states with dependent parameters (analogously to lower and upper limit with numeric parameters).

NrpGetBitfieldResult reads from the bit-field buffer the oldest bit-field blocks that have been sent since the function was last called by the sensor. The bit-field block read is deleted from the bit-field buffer. The bit fields are transferred as *LONG* variables.

The function returns 1 if read-out was successful and 0 if an error occurred during read-out.

Prototype:

`long NrpGetBitfieldResult (long *o_pdwR1, long *o_pdwR2, long *o_pdwR3);`

Parameters: *o_pdwR1* Pointer to the first of the three *LONG* variables that have to store the result.
o_pdwR2 Pointer to the second of the three *LONG* variables that have to store the result.
o_pdwR3 Pointer to the third of the three *LONG* variables that have to store the result.

NrpGetFloatArray

If the sensor is in the *buffered mode*, the results are transmitted in blocks, not individually. A particularly high transmission rate can thus be obtained. The sensor also returns its results in the form of *FLOAT* array blocks in the *Timeslot* and *Scope* modes. A fixed index is assigned to each *FLOAT* value in the array. *NrpGetFloatArray* copies a defined number of *FLOAT* values and their indices starting with the oldest values, from the DLL-internal float array buffer to a *FLOAT* or *LONG* array. The copied values are deleted from the float array buffer. The function returns the number of actually copied *FLOAT* values or indices.

If the user transfers a zero pointer as *o_pFarr* argument to the function, the function supplies the number of *FLOAT* values and indices contained in the float array buffer.

Prototype:

```
long NrpGetFloatArray (float *o_pFarr , long *o_pIarr, long i_wCount);
```

Parameters:

<i>o_pFarr</i>	Pointer to the <i>FLOAT</i> array that has to store the <i>FLOAT</i> values
<i>o_plarr</i>	Pointer to the <i>LONG</i> array that has to store the indices of <i>FLOAT</i> values
<i>i_wCount</i>	Length of array

NrpGetFloatResult

NrpGetFloatResult copies the three *FLOAT* values contained in a *FLOAT* block to a *FLOAT* variable. If the *FLOAT* block is a measurement result, the first *FLOAT* value is the power value; the second and third *FLOAT* values are secondary measured values, e.g. noise and measurement uncertainty. The function returns 1 if read-out was successful and 0 if an error occurred during read-out.

Prototype:

```
long NrpGetFloatResult(float *o_pfR1, float *o_pfR2, float *o_pfR3);
```

Parameters:

<i>o_pfR1</i>	Pointer to the <i>FLOAT</i> variable that has to store the first <i>FLOAT</i> value
<i>o_pfR2</i>	Pointer to the <i>FLOAT</i> variable that has to store the second <i>FLOAT</i> value
<i>o_pfR3</i>	Pointer to the <i>FLOAT</i> variable that has to store the third <i>FLOAT</i> value

NrpGetLongResult

NrpGetFloatResult copies the three *LONG* values contained in a *LONG* block to a *FLOAT* variable. The function returns 1 if read-out was successful and 0 if an error occurred during read-out.

Prototype:

```
long NrpGetLongResult (long *o_pwR1, long *o_pwR2, long *o_pwR3);
```

Parameters:

<i>o_pwR1</i>	Pointer to the <i>LONG</i> variable that has to store the first <i>LONG</i> value
<i>o_pwR2</i>	Pointer to the <i>LONG</i> variable that has to store the second <i>LONG</i> value
<i>o_pwR3</i>	Pointer to the <i>LONG</i> variable that has to store the third <i>LONG</i> value

NrpGetStringResult

NrpGetStringResult enables access to the string responses received until the function is called. A string response is usually received in several blocks, reassembled in the DLL and stored in the internal string buffer. The function copies a defined number of characters from this internal string buffer to a buffer. The copied characters are deleted from the internal string buffer. The function returns the number of characters actually read.

If the user transfers a zero pointer as argument *o_pBuf* to the function, the function returns the number of characters contained in the string buffer. The terminating zero character is not counted. If the terminating zero character is to be read out, a buffer that is larger by 1 should be transferred.

Prototype: `long NrpGetStringResult (char *o_pBuf, long i_wCount);`

Parameters:

<i>o_pBuf</i>	Pointer to the buffer in which the response has to be stored
<i>i_wCount</i>	Buffer size in bytes

Functions for Status and Error Detection

NrpEmptyAllBuffers

NrpEmptyAllBuffers deletes all DLL-internal dynamic buffers.

Prototype: `void NrpEmptyAllBuffers (void);`

NrpEmptyErrorQueue

NrpEmptyErrorQueue deletes all errors contained in the error queue.

Prototype: `void NrpEmptyErrorQueue (void);`

NrpGetLastError

NrpGetLastError reads the oldest errors from the error queue. To empty the error queue, either all errors must be read out consecutively with *NrpGetLastError* or the function *NrpEmptyErrorQueue* or *NrpEmptyAllBuffers* must be called. A list of all possible error states is provided in Table 5-4.

Prototype: `long NrpGetLastError (void);`

Table 5-4 Possible sensor error states

Error (enum constant)	Ordinal number	Meaning
NRPEERROR_NOERROR	0	This value is returned by the function <i>NrpGetLastError</i> if no error occurred.
NRPEERROR_CALDATA_FORMAT	1	The format of the calibration data set to be transmitted to the sensor does not comply with the specification. Either the available calibration data set is not recognized as such or its version is not supported.
NRPEERROR_OVERRANGE	2	The power of the test signal exceeds the range of the measurement channel selected manually (with <i>SENSe:RANGe:AUTO OFF</i>).
NRPEERROR_NOTINSERVICEMODE	3	The command sent to the sensor is only available in the service mode but the sensor is not set to the service mode.
NRPEERROR_CALZERO	4	Zeroing could not be performed because the RF power applied is too high.
NRPEERROR_TRIGGERQUEUEFULL	5	In the <i>Burst Average</i> mode: The burst is too large and, as a result, the measurement was interrupted after the maximum permissible burst duration elapsed. The result thus obtained may deviate from the correct measured value.
NRPEERROR_EVENTQUEUEFULL	6	The internal queue for trigger events is full. This error message indicates a software error since the sensor usually quits accepting trigger events when an overflow is expected.
NRPEERROR_SAMPLEERROR	7	One or several samples could not be processed because the sensor did not react to the sample interrupt during the processing of a bus trigger, for example. Normally, this corrupts the measurement result, except in the realtime mode (<i>SENSe:SWEep:REALtime ON</i>).

Error (enum constant)	Ordinal number	Meaning
NRPEROR_OVERLOAD	8	The power of the test signal exceeds the upper measurement limit of the sensor. The sensor may irreversibly be damaged.
NRPEROR_HARDWARE	9	A hardware error has occurred. Errors on the temperature sensor immediately generate this error message during operation. If an error was detected with test command <i>*TST?</i> or <i>TEST:SENsOr?</i> , the error message will be output until a new selftest detects no error. Errors in operating voltages are signalled separately (refer to <i>NRPEROR_VOLTAGE</i>).
NRPEROR_CHECKSUM	10	The checksum verification of the calibration data set to be loaded generated an error. The calibration data set was not loaded.
NRPEROR_ILLEGALSERIAL	11	An attempt was made to load a calibration data set whose serial number does not correspond to the serial number of the sensor. The calibration data set was not loaded.
NRPEROR_FILTERTRUNCATED	12	In the Auto-Averaging mode: The measurement was interrupted after the maximum measurement time (parameter <i>SENSe:AVERAge:COUNt:AUTO:MTIME</i> or <i>SENSe:SWEEp:AVERAge:COUNt:AUTO:MTIME</i>) without obtaining the defined resolution or the defined signal/noise ratio.
NRPEROR_GENERIC	128	An error that cannot be defined more precisely has occurred.
NRPEROR_OVERMAX	129	The numeric parameter is greater than the permissible upper limit.
NRPEROR_UNDERMIN	130	The numeric parameter is smaller than the permissible lower limit.
NRPEROR_VOLTAGE	131	At least one of the internal supply voltages is not applied or is out of the permissible range.
NRPEROR_SYNTAX	132	The syntax of the command sent to the sensor is erroneous.
NRPEROR_MEMORY	133	The memory available is not sufficient. This error message indicates a firmware error.
NRPEROR_PARAMETER	134	This parameter is not allowed.
NRPEROR_TIMING	135	reserved
NRPEROR_NOTIDLE	136	The command sent to the sensor is not available during a measurement.
NRPEROR_UNKNOWNCOMMAND	137	The command sent to the sensor could not be interpreted.
NRPEROR_OUTBUFFERFULL	138	The transmit buffer of the sensor has overflowed because the data output by the sensor were not fetched.
NRPEROR_FLASHPROG	139	A flash memory programming error has occurred.
NRPEROR_CALDATANOTPRESENT	140	No valid calibration data set is present.

NrpGetErrorText

NrpGetErrorText calls the error message in plain text corresponding to the error code of the *NRPEERROR* type. This message is stored as a non-zero-terminated string in a buffer. The function returns the number of characters stored in the buffer.

If the user transfers a zero pointer as argument *o_pBuf* to the function, the function returns the number of characters contained in the string buffer. The terminating zero character is not counted. If the terminating zero character is to be read out, a buffer that is larger by 1 should be transferred.

Prototype: `long NrpGetErrorText (long i_wErr, char *o_pBuf, long i_wCount);`

Parameters:

<i>i_wErr</i>	Error code that was supplied by <i>NrpGetLastError</i> , for example
<i>o_pBuf</i>	Pointer to the buffer in which the error text is to be stored
<i>i_wCount</i>	Buffer size in bytes

NrplsAlive

NrplsAlive returns

- 0 if the selected active sensor is not ready for operation
- 1 if it is ready for operation and in the measurement mode
- 2 if the boot loader is active.

Prototype: `long NrpIsAlive (void);`

NrpGetTriggerState

NrpGetTriggerState returns the consecutive number of the current trigger system state (Table 5-5). The function *NrpGetTriggerStateText* can be used to obtain a short description in plain text of the trigger state.

Prototype: `long NrpGetTriggerState (void);`

Table 5-5 Possible sensor trigger states

Trigger state (enum constant)	Ordinal number	Description
SENSORHW_STATE_IDLE	0	No measurement is in progress and the sensor is ready for operation.
SENSORHW_STATE_WAIT_FOR_ARM	1	Reserved. The R&S NRP-Z11/-Z21 has no ARM layer.
SENSORHW_STATE_WAIT_FOR_TRIGGER	2	A measurement was started and the sensor is waiting for a trigger event.
SENSORHW_STATE_MEASURING	3	A measurement is in progress.

NrpGetTriggerStateText

NrpGetTriggerStateText provides a short description in plain text of the trigger state determined with *NrpGetTriggerState*, for example.

If the user transfers a zero pointer as argument *o_pBuf* to the function, the function returns the number of characters in the descriptive string. The terminating zero character is not counted. If the terminating zero character is to be read out, a buffer that is larger by 1 should therefore be transferred.

Prototype:

```
long NrpGetTriggerStateText (long i_wState, char *o_pBuf, long i_wCount);
```

Callback Functions

Callback functions that can be called by the DLL when relevant events occur can be defined to allow immediate and asynchronous evaluation of these events by the application. The following events can generate a response:

- *CommandAccepted*: A command sent to the sensor was accepted by the sensor.
- *DataAvailable*: Data from the sensor is available.
- *DeviceChanged*: A sensor was connected to the host controller or disconnected from the host controller.
- *ErrorOccurred*: The sensor has signalled an error.
- *StateChanged*: The trigger state of the sensor has changed.

The following functions are used to defined callback functions:

- *NrpSetNotifyCallbackCommandAccepted*
- *NrpSetNotifyCallbackDataAvailable*
- *NrpSetNotifyCallbackDeviceChanged*
- *NrpSetNotifyCallbackErrorOccurred*
- *NrpSetNotifyStateChanged*

These functions are described in detail in section "Functions for Setting Callback Functions" on page 5.19. The *function* type is defined as follows:

```
typedef void (__stdcall *function) (void *arg1, void *arg2);
```

The callback function to be transmitted should have the following form:

```
void __stdcall MyCallbackFunction (void *arg1, void *arg2)
{
    ...
}
```

If the callback function is a class method it must be defined as *static*.

With events *CommandAccepted*, *DataAvailable*, *ErrorOccured* and *StateChanged*, the argument *arg1* is assigned *NULL* and need not be evaluated any longer. With *DeviceChanged*, a pointer is transmitted to a structure of the *USB_DEVICECHANGED_HANDLER_RESULT* type (see description of the function *NrpSetNotifyCallbackDeviceChanged* on page 5.19) that can be evaluated for differentiated event processing.

The argument *arg2* is used to transmit specific context information (e.g. reference to a class if the callback function should call a method of this class). The argument is specified on calling one of the *SetNotifyCallback* functions.

The following example shows a callback function for the *DeviceChanged* event that is implemented as class method in Visual C++. This is not a program that can be automatically executed. This applies in particular to class method *AddToProtocol* not presented here for outputting a string in a multiline edit control, for example.

Definition in the header file:

```
private:
    static void __stdcall DeviceChangedCallback(void* arg1, void* arg2);
```

Registration in the initialization section of the class:

```
NrpSetNotifyCallbackDeviceChanged(DeviceChangedCallback, this);
```

Implementation:

```
void CTestDlg::DeviceChangedCallback(void* arg1, void* arg2)
{
    CString txt, name, msg;
    long length, devices, devID, aliveState;

    USB_DEVICECHANGED_HANDLER_RESULT* hRes =
        (USB_DEVICECHANGED_HANDLER_RESULT*) arg1;

    CTestDlg* dialog = (CTestDlg*) arg2;

    length = NrpGetDeviceChangedMsgText(hRes->wEventTextHandle, NULL, 0);
    NrpGetDeviceChangedMsgText(hRes->wEventTextHandle, txt.GetBuffer(length),
        length);
    msg = "DeviceChangedMsgText:\t" + txt + "\r\n";
    dialog->AddToProtocol(msg);

    length = NrpGetDeviceChangedDevName(NULL, 0);
    NrpGetDeviceChangedDevName(name.GetBuffer(length), length);
    msg = "DeviceChangedDevName:\t" + name + "\r\n";
    dialog->AddToProtocol(msg);

    msg.Format("t\t%i Sensor(s) connected\r\n", NrpGetNrOfDevices());
    dialog->AddToProtocol(msg);
}
```

Functions for Setting Callback Functions

NrpSetNotifyCallbackCommandAccepted

NrpSetNotifyCallbackCommandAccepted defines a user-specific function that is called if a command sent to the sensor was successfully processed and acknowledged by the sensor (see section "Callback Functions" on page 5.17).

Prototype: void NrpSetNotifyCallbackCommandAccepted (function f, void *arg);

Parameters: *f* Function to be called
 arg Argument for transferring context information

NrpSetNotifyCallbackDataAvailable

NrpSetNotifyCallbackDataAvailable defines a user-specific function that is called if measured values or other data from the sensor are available for retrieval (see section "Callback Functions" on page 5.17).

Prototype: void NrpSetNotifyCallbackDataAvailable (function f, void *arg);

Parameters: *f* Function to be called
 arg Argument for transferring context information

NrpSetNotifyCallbackDeviceChanged

NrpSetNotifyCallbackDeviceChanged defines a user-specific function that is called if a sensor is connected to the host controller or disconnected from the host controller (see section "Callback Functions" on page 5.17).

Prototype: void NrpSetNotifyCallbackDeviceChanged (function f, void *arg);

Parameters: *f* Function to be called
 arg Argument for transferring context information

NrpSetNotifyCallbackErrorOccured

NrpSetNotifyCallbackErrorOccured defines a user-specific function that is called if an error occurred (see section "Callback Functions" on page 5.17).

Prototype: void NrpSetNotifyCallbackErrorOccured (function f, void *arg);

Parameters: *f* Function to be called
 arg Argument for transferring context information

NrpSetNotifyCallbackStateChanged

NrpSetNotifyCallbackErrorOccured defines a user-specific function that is called if the trigger state of the sensor has changed (see section "Callback Functions" on page 5.17).

Prototype: void NrpSetNotifyCallbackStateChanged (function f , void *arg);

Parameters: *f* Function to be called
 arg Argument for transferring context information

Windows Messages

The DLL sends and receives Windows™ messages. It requires that the calling program transfers these messages. The calling program is usually a GUI application based on the Windows API so that this condition is always fulfilled. However, if the program is a simple console application, this application will transfer no messages. The DLL offers the function *NrpMessageLoopBody* to add a message loop in a console application. It should be cyclically called in the main loop of the console application to transfer the Windows™ messages sent and received by the DLL.

Prototype: `void NrpMessageLoopBody (void);`

USB-specific Commands

The following commands only concern the USB interface and do not refer to the sensor functionality.

NrpOpenDriver

NrpOpenDriver establishes the connection to the USB device driver.

Prototype: `void NrpOpenDriver (void);`

NrpCloseDriver

NrpCloseDriver terminates the connection to the USB device driver.

Prototype: `void NrpCloseDriver (void);`

Note: When the DLL is closed, any connection to the USB device driver is automatically cleared down.

NrpClearDevice

NrpClearDevice sends the *Vendor Request SET_DEVICE_CLEAR* to the sensor. This *Vendor Request* deletes the USB FIFOs and initializes the trigger system.

Prototype: `void NrpClearDevice (void);`

NrpGetDeviceChangedDevName

The callback function that is called by the *DeviceChanged* event supplies the structure *USB_DEVICECHANGED_HANDLER_RESULT* as an argument (see Table 8). One of the structure element is the pointer *pDeviceName* to the name of the device that has triggered the event. Since strings cannot be transferred directly as pointers in specific programming environments such as Visual Basic, the function *NrpGetDeviceChangedDevName* is offered as an alternative that copies the device name to a buffer. The function returns the number of actually copied characters.

If the user transfers a zero pointer as argument *o_pBuf* to the function, the function returns the number of characters in the device-name string. The terminating zero character is not counted. If the terminating zero character is to be read out, a buffer that is larger by 1 should be transferred.

Prototype: `long NrpGetDeviceChangedDevName (char *o_pBuf, long i_wCount);`

Parameters: *o_pBuf* Pointer to the buffer that has to store the device name
i_wCount Buffer size in bytes

Table 5-6 Description of structure *USB_DEVICECHANGED_HANDLER_RESULT*

Type	Name	Description
long	wResult	The value is 1 if a device was assigned to the event and 0 if this is not the case (the string to which pDeviceName points is empty).
long	wEventType	Can take the values <i>SENSORHW_STATE_DEVICEARRIVAL</i> to <i>SENSORHW_STATE_DEVICEUNKNOWN</i> (see Table 5-7).
long	wEventTextHandle	Text handle that enables the function <i>NrpGetDeviceChangedMsgText</i> to determine the plain-text description of the event.
char*	pEventText	Pointer to the plain-text description of the event. This pointer should only be used in a C or C++ environment. The functions <i>NrpGetDeviceChangedMsgText</i> and <i>NrpGetDeviceChangedMsgTextLength</i> must be used for Visual Basic.
char*	pDeviceName	The device name that has triggered the event. This pointer should only be used in a C or C++ environment. The functions <i>NrpGetDeviceChangedDevName</i> and <i>NrpGetDeviceChangedDevNameLength</i> must be used for Visual Basic.

NrpGetDeviceChangedMsgText

The callback function that is called by the *DeviceChanged* event returns the structure *USB_DEVICECHANGED_HANDLER_RESULT* as an argument (see Table 5-6). One of the structure element is *wEventTextHandle*, a *LONG* value behind which a DLL-internal string-list index is concealed. *NrpGetDeviceChangedMsgText* is used to copy the event message stored in the string list to a buffer. The function returns the number of actually copied characters.

If the user transfers a zero pointer as argument *o_pBuf* to the function, the function returns the number of characters in the event message. The terminating zero character is not counted. If the terminating zero character is to be read out, a buffer that is larger by 1 should be transferred.

Prototype:

`long NrpGetDeviceChangedMsgText (long wEventTextHandle, char *o_pBuf, long i_wCount);`

Parameters: *wEventTextHandle* String list index
o_pBuf Pointer to the buffer that has to store the event message
i_wCount Buffer size in bytes

NrpGetDeviceID

NrpGetDeviceID returns the consecutive number of the active sensor. Each connected sensor is assigned such a consecutive number. Counting starts with 0.

Prototype: `long NrpGetDeviceID (void);`

NrpGetIDByLogicalName

Each sensor has a unique combination of type designation and serial number that is automatically assigned a logical name when it is connected to the USB for the first time. These assignments are entered in the registry under *HKEY_CURRENT_USER\Software\Rohde&Schwarz\NrpDII\Sensors*. They can be modified by means of function *NrpChannelAssignment*. *NrpGetIDByLogicalName* returns the consecutive number of a connected sensor if the transferred logical name was found in the registry. If the logical name was not found in the registry or if the respective sensor is not connected, the function returns -1.

Prototype: `long NrpGetIDByLogicalName (const char *i_pLogicalName);`

Parameters: *i_pLogicalName* Logical name of the sensor as zero-terminated string

NrpGetLogicalName

Each sensor has a unique combination of type designation and serial number that is automatically assigned a logical name when it is connected to the USB for the first time. These assignments are entered in the registry under *HKEY_CURRENT_USER\Software\Rohde&Schwarz\NrpDII\Sensors*. They can be modified by means of function *NrpChannelAssignment*. *NrpGetLogicalName* copies the logical name of the connected sensor with the consecutive number transferred as an argument to a buffer. The function returns the number of actually copied characters.

If the user transfers a zero pointer as argument *o_pBuf* to the function, the function returns the number of characters in the logical name. The terminating zero character is not counted. If the terminating zero character is to be read out, a buffer that is larger by 1 should be transferred.

Prototype: `long NrpGetLogicalName (char *o_pBuf, long i_wCount, long i_wID);`

Parameters:

<i>o_pBuf</i>	Pointer to the buffer that has to store the logical name
<i>i_wCount</i>	Buffer size in bytes
<i>i_wID</i>	Consecutive number of the sensor

NrpGetManufacturerCode

NrpGetManufacturerCode copies the manufacturer name of the connected sensor with the consecutive number transferred as an argument to a buffer. The function returns the number of actually copied characters.

If the user transfers a zero pointer as argument *o_pBuf* to the function, the function returns the number of characters in the logical name. The terminating zero character is not counted. If the terminating zero character is to be read out, a buffer that is larger by 1 should be transferred.

Prototype:

```
long NrpGetManufacturerCode(char *o_pBuf, long i_wCount, long i_wID);
```

Parameters:

<i>o_pBuf</i>	Pointer to the buffer that has to store the manufacturer name
<i>i_wCount</i>	Buffer size in bytes
<i>i_wID</i>	Consecutive number of the sensor

NrpGetNrOfDevices

NrpGetNrOfDevices returns the number of sensors connected to the USB.

Prototype: long NrpGetNrOfDevices (void);

NrpGetProductID

NrpGetProductID copies the type designation of the connected sensor with the consecutive number transferred as an argument to a buffer. The function returns the number of actually copied characters.

If the user transfers a zero pointer as argument *o_pBuf* to the function, the function returns the number of characters in the type designation. The terminating zero character is not counted. If the terminating zero character is to be read out, a buffer that is larger by 1 should therefore be transferred.

Prototype:

```
long NrpGetProductID(char *o_pBuf, long i_wCount, long i_wID);
```

Parameters:

<i>o_pBuf</i>	Pointer to the buffer that has to store the type designation
<i>i_wCount</i>	Buffer size in bytes
<i>i_wID</i>	Consecutive number of the sensor

NrpGetSerialNumber

NrpGetSerialNumber copies the serial number string of the connected sensor with the consecutive number transferred as an argument to a buffer. The function returns the number of actually copied characters.

If the user transfers a zero pointer as argument *o_pBuf* to the function, the function returns the number of characters in the serial number string. The terminating zero character is not counted. If the terminating zero character is to be read out, a buffer that is larger by 1 should therefore be transferred.

Prototype:

```
long NrpGetProductID(char *o_pBuf, long i_wCount, long i_wID);
```

Parameters:

<i>o_pBuf</i>	Pointer to the buffer that has to store the serial number string
<i>i_wCount</i>	Buffer size in bytes
<i>i_wID</i>	Consecutive number of the sensor

NrpSDRGetDescriptor

NrpSDRGetDescriptor sends the *Standard Device Request GetDescriptor* to the sensor. This *Standard Device Request* is used to read device, configuration and string descriptors. Interface and endpoint descriptors cannot be read out directly but only together with the configuration descriptor. The function returns the error status according to Table 5-4. The structure of descriptors is described in detail in the technical literature on USB.

Prototype:

```
long NrpSDRGetDescriptor (void *o_pBuffer, long *io_wByteCount,
                          long i_wRecipient, long i_wDescrType,
                          long i_wDescrIndex, long i_wLangID);
```

Parameters:	<i>o_pBuffer</i>	Pointer to the buffer that has to store the descriptor data
	<i>io_wByteCount</i>	After being called: Buffer size in bytes After being called: Number of bytes actually stored in the buffer
	<i>i_wRecipient</i>	0 = device, 1 = interface, 2 = endpoint, 3 = other
	<i>i_wDescrType</i>	Type of descriptor (1 = device descriptor, 2 = configuration descriptor incl. associated interface, class and endpoint descriptors, 3 = string descriptor)
	<i>i_wDescrIndex</i>	Only with string descriptor: Index of the string for manufacturer name, product name or series number
	<i>i_wLangID</i>	Only with string descriptor: Language index (optional, 0x0409 = English), otherwise 0

NrpSelectDevice

NrpSelectDevice selects the connected sensor with the consecutive number transferred as an argument.

Prototype: void NrpSelectDevice (long i_wID);

Parameter: *i_wID* Consecutive number of the sensor

Examples of Application

The following examples illustrate the sensor functionality. The program module **USB-Terminal** of the NRP toolkit is suitable for learning the commands. The sensor is ready for operation immediately after it has been connected to the USB connector of the PC.

Commands must be sent to the sensor by means of the function *NrpSendCommand*. For clarity, only the SCPI command sequences are shown below. Only the short form of commands is used. For more details on the commands, refer to the command reference in section 6 of this user manual.

Before the measurement zeroing should be performed with the sensor with power switched off.

```
cal:zero:auto once          or          cal:zero:auto on
```

In addition, the RF carrier frequency (500 MHz in the following example) should be reported to the sensor for each measurement.

```
sens:freq 500e6
```

Single Average Power Measurement

In the simplest case, only the following commands are required to measure the average power:

```
*rst          resets the sensor to the default settings  
init:imm      starts the measurement
```

An averaging filter is provided in all measurement modes. This filter can be operated with an averaging factor that is either manually set or automatically determined by the sensor. If the default setting is used, the sensor automatically determines the averaging factor. Two algorithms can be selected to determine the averaging factor. With the classic method, the required resolution is preset and the sensor determines the filter length using the power applied and the upper limit for the measurement time is complied with (parameter *SENSe:AVERage:COUNt:AUTO:MTIME*). With the new fixed-noise method (*SENSe:AVERage:COUNt:AUTO:TYPE NSRatio*), however, an exactly defined noise component is complied with in the result.

The averaging filter can operate in two different ways. The parameter *SENSe:AVERage:TCONtrol* can accept the values *MOVing* and *REPeat*.

- The *REPeat* setting is usually of advantage in the remote-control mode. The average value is output when the filter is completely filled. In the *Continuous Average* mode, only one trigger event is required to start the measurement for filling the averaging filter; the other measurements are automatically performed. In the modes *Burst Average*, *Timeslot* and *Scope*, one trigger event is required for each measured value because of the time reference.
- With the *MOVing* setting, the average value is formed and output each time a measured value is shifted into the filter. This filter behaviour allows detection at an early stage of a trend in the displayed average value. This behaviour is also used in manual operation via the basic unit.

Example of configuration with manual averaging and *REPeating* behaviour:

```
sens:aver:coun:auto off
sens:aver:coun 8
sens:aver:tcon rep
```

Example of configuration with auto-averaging and fixed-noise method (noise component: 0.01 dB, upper limit for measuring time: 30 s):

```
sens:aver:coun:auto on
sens:aver:coun:auto:type nsr
sens:aver:coun:auto:nsr 0.01
sens:aver:count:auto:mtim 30
```

The averaging filter settings are the same for the modes *Continuous Average*, *Burst Average* and *Timeslot* and are accepted if the mode changes. The filter is specially configured for the *Scope* mode; the commands begin with *SENSe:SWEp:AVERage*: ...

Quick and Multiple Average Power Measurement

For quick successive measurements, the sensor provides a result buffer with settable size. The content of this buffer is transferred to the control unit (basic unit/PC) after the buffer is filled with measured values. Since the content is transferred in *FLOAT* array blocks, the transmission is particularly time-saving. The following example shows the settings for the quickest possible operation:

*rst	resets the sensor to default settings
sens:freq 500e6	sets the RF carrier frequency (here 500 MHz)
sens:aver:stat off	switches off the averaging filter
sens:pow:avg:aper 100e-6	sets the shortest measurement window (here 100 μ s)
sens:pow:avg:buff:size 100	the result buffer has to store 100 measured values
sens:pow:avg:buff:stat on	switches on the result buffer
trig:sour imm	free-running trigger
trig:coun 100	stores 100 measured values
init:imm	starts the measurement

The command `trig:count 100` ensures that 100 measurements are performed after a single measurement start with `init:imm` to fill the buffer. If, however, the default setting `trig:count 1` is used, `init:imm` would have to be called 100 times successively for the same number of results.

Measuring a Single Burst

In the *Burst Average* mode, the sensor can automatically detect individual bursts and measure their average power value. In this measurement mode, the sensor waits for a rising trigger edge and continues measuring until it recognizes a falling edge. If no burst end is detected within 50 ms, the sensor interrupts the measurement and outputs the error message *NRPError_TRIGGERQUEUEFULL*.

*rst	resets the sensor to default settings
sens:func "pow:burs:avg"	sets the <i>Burst Average</i> mode
trig:lev 1e-6	sets the trigger threshold to 1 μ W
sens:pow:burs:dtol 50e-6	sets the dropout tolerance
init:imm	starts the measurement

In addition, sections at the burst beginning and end can be excluded (e.g. to eliminate the influence of flat signal edges or overshoots on the measurement result). In the following example, the first 20 μs and the last 30 μs of the burst are excluded from the measurement:

```
sens:tim:excl:star 20e-6
sens:tim:excl:stop 30e-6
```

Measurement of a Frame with Several Timeslots

The sensor provides the *Timeslot* mode for measuring the power in several successive timeslots in a fixed time frame. The number and duration of timeslots have to be set for the measurement. A measurement is triggered by the test signal (*TRIGger:SOURce INTernal*) or by an externally applied trigger signal (*TRIGger:SOURce EXTernal*). An external trigger source is preferable to avoid measurement errors due to inaccurate triggering. Triggering via the USB (*TRIGger:SOURce BUS*) is basically possible, but it is not recommended because of the inaccurate time resolution.

The following example shows how the power can be measured in the eight timeslots of a GSM/EDGE signal:

```
*rst                resets the sensor to default settings
sens:func "pow:tsl:avg"  selects the Timeslot mode
sens:freq 500e6        sets the RF carrier frequency (here 500 MHz)
sens:aver:coun:auto off  deactivates auto-averaging
sens:aver:coun 8        sets averaging factor 8
trig:sour ext          selects an external trigger source (int can also be entered for
                        internal triggering; the trigger threshold should then be defined
                        with TRIGger:LEVel.)

trig:slop pos         triggering to the rising edge of the trigger signal
sens:pow:tsl:avg:coun 8  number of successive timeslots (here 8)
sens:pow:tsl:avg:widt 570e-6  width of a timeslot in seconds (here 570  $\mu\text{s}$  = width of a
                                GSM timeslot)

init:imm             starts the measurement
```

As in the *Burst* mode, the start and end sections of timeslots can be excluded from the measurement in the *Timeslot* mode. The setting parameters are the same as in the *Burst Average* mode:

```
sens:tim:excl:star 50e-6    the first 50  $\mu\text{s}$  of each timeslot are blanked
sens:tim:excl:stop 80e-6    the last 80  $\mu\text{s}$  of each timeslot are blanked
```



The parameters *SENSe:TIMing:EXCLude:STARt* and *STOP* are valid for both the *Burst Average* mode and the *Timeslot* mode, i.e. the settings performed in either of the two modes are accepted in the other mode when the mode changes.

Scope Mode for Displaying Periodic Signals

The sensor provides the *Scope* mode to display the time sequence of power in the same manner as with an oscilloscope. As in the *Timeslot* mode, triggering can be performed by the test signal or an externally applied trigger signal. An external trigger source is also preferable to avoid measurement errors due to inaccurate triggering.

The *Scope* mode provides separate parameters for the averaging filter.

To configure the sensor for a *Scope* measurement, the time to be covered by the *Scope* presentation and the number of intervals in which the presentation is to be resolved have to be specified. The smallest time resolution of the sensor is 10 μ s. Ensure that the interval width (= time/number of intervals) does not exceed this time when setting the above parameters since otherwise the *Scope* presentation is corrupted.

The following example shows the use of the *Scope* mode, the trigger being derived from the test signal:

```
*rst                resets the sensor to default settings
sens:func "xtim:pow"  selects the Scope mode
sens:freq 500e6      sets the RF carrier frequency (here 500 MHz)
sens:swe:aver:coun:auto off  deactivates auto-averaging
sens:swe:aver:coun 8    sets averaging factor 8
trig:sour int        selects triggering by the test signal
trig:slop pos        triggering to the rising edge of the test signal
trig:lev 1e-6        sets the trigger threshold (here 1  $\mu$ W)
trig:hyst 3          sets the trigger hysteresis (here 3 dB, i.e. new triggering due
                    to the trigger threshold being exceeded is only possible if
                    the test signal level has underranged the trigger threshold
                    by at least 3 dB)
sens:swe:time 20e-3   time in seconds to be covered by the Scope presentation
                    (here 20 ms)
sens:swe:poin 200    number of intervals in which the Scope presentation is to be
                    resolved (here 200, i.e. resolution = 20 ms / 200 = 100  $\mu$ s)
init:imm            starts the measurement
```

Scope Mode for Displaying a Single Process (Single-Shot Measurement)

A single non-periodic signal can be detected in the *Scope* mode if the parameter *SENSe:SWEp:REALtime ON* is set. Averaging is of course not possible with single processes. The averaging filter settings are therefore ignored (and not modified). In contrast to the last two modes, bus triggering is possible here since averaging does not cause several measurements to be superimposed.

The following example shows the settings for the single-shot measurements:

```
*rst                resets the sensor to default settings
sens:func "xtim:pow"  selects the Scope mode
sens:freq 500e6      sets the RF carrier frequency (here 500 MHz)
sens:swe:time 20e-3   time in seconds to be covered by the Scope presentation
                    (here 20 ms)
sens:swe:poin 200    number of intervals in which the Scope presentation is to be
                    resolved (here 200, i.e. resolution = 20 ms / 200 = 100  $\mu$ s)
sens:swe:real on     activates the single-shot mode
trig:sour ext        selects an external trigger source (int can also be entered for
                    internal triggering; the trigger threshold should then be defined
                    with TRIGger:LEVel.)
trig:slop pos        triggering to the rising edge of the trigger signal
init:imm            starts the measurement
```


The following commands are required for a bus-triggered single-shot measurement:

<code>trig:sour hold</code>	ignores all trigger events, except <i>TRIGger:IMMEDIATE</i>
<code>init:imm</code>	initializes the measurement; the trigger system waits for a trigger event
<code>trig:imm</code>	triggers the single-shot measurement

Note that the trigger delay (parameter *TRIGger:DELay*) is ignored due to the command *TRIGger:IMMEDIATE*.

Table of Contents

6 Remote Control – Commands	6.1
Notation	6.1
Commands as per IEEE 488.2	6.2
*IDN? – Identification Query	6.2
*RST – Reset	6.2
*TRG – Trigger	6.2
*TST? – Self Test Query	6.2
SCPI Commands	6.3
CALibration.....	6.3
CALibration:DATA[?] <calibration data set as <i>definite length block</i> >	6.3
CALibration:DATA:LENGth?	6.3
CALibration:ZERO:AUTO[?] OFF ON ONCE	6.4
SENSe (Sensor Configuration)	6.5
SENSe:AVERAge:COUNt[?] 1 to 65536	6.7
SENSe:AVERAge:COUNt:AUTO[?] OFF ON ONCE	6.7
SENSe:AVERAge:COUNt:AUTO:MTIME[?] 1.0 to 999.99.....	6.7
SENSe:AVERAge:COUNt:AUTO:NSRatio[?] 0.0001 to 1.0	6.8
SENSe:AVERAge:COUNt:AUTO:RESolution[?] 1 to 4.....	6.8
SENSe:AVERAge:COUNt:AUTO:SLOT[?] 1 to <SENSe:POWer:TSLot:AVG:COUNt>.....	6.8
SENSe:AVERAge:COUNt:AUTO:TYPE[?] RESolution NSRatio	6.8
SENSe:AVERAge:STATe[?] OFF ON.....	6.9
SENSe:AVERAge:TCONtrol[?] MOVing REPeat.....	6.9
SENSe:CORRection:DCYClE[?] 0.001 to 99.999	6.9
SENSe:CORRection:DCYClE:STATe[?] OFF ON	6.9
SENSe:CORRection:OFFSet[?] –100.0 to 100.0	6.10
SENSe:CORRection:OFFSet:STATe[?] OFF ON.....	6.10
SENSe:CORRection:SPDev:STATe[?] OFF ON	6.10
SENSe:FREQuency[?] 0.0 to 100.0e9.....	6.10
SENSe:FUNcTION[?] <sensor_function>	6.11
SENSe:POWer:AVG:APERture[?] 0.0001 to 0.1	6.12
SENSe:POWer:AVG:BUFFer:SIZE[?] 1 to 1024	6.12
SENSe:POWer:AVG:BUFFer:STATe[?] OFF ON.....	6.12
SENSe:POWer:AVG:SMOothing:STATe[?] OFF ON	6.13
SENSe:POWer:BURSt:DTOLerance[?] 0.0 to 0.003.....	6.13
SENSe:POWer:TSLot:AVG:COUNt[?] 1 to 128.....	6.14
SENSe:POWer:TSLot:AVG:WIDTh[?] 0.0001 to 0.1	6.14
SENSe:RANGe[?] 0 to 2	6.14
SENSe:RANGe:AUTO[?] OFF ON	6.15
SENSe:RANGe:AUTO:CLEVel[?] –20.0 to 0.0.....	6.15
SENSe:SAMPLing[?] FREQ1 FREQ2.....	6.15
SENSe:SGAMma:CORRection:STATe[?] OFF ON.....	6.15
SENSe:SGAMma:MAGNitude[?] 0.0 to 1.0	6.15
SENSe:SGAMma:PHASe[?] –360.0 to 360.0.....	6.16

SENSe:SWEEp:AVERAge:COUNT[?] 1 to 65536.....	6.16
SENSe:SWEEp:AVERAge:COUNT:AUTO[?] OFF ON ONCE.....	6.16
SENSe:SWEEp:AVERAge:COUNT:AUTO:MTIME[?] 1.0 to 999.99	6.16
SENSe:SWEEp:AVERAge:COUNT:AUTO:NSRatio[?] 0.0001 to 1.0	6.17
SENSe:SWEEp:AVERAge:COUNT:AUTO:RESolution[?] 1 to 4.....	6.17
SENSe:SWEEp:AVERAge:COUNT:AUTO:POINT[?] 1 to <SENSe:SWEEp:POINTs>.....	6.17
SENSe:SWEEp:AVERAge:COUNT:AUTO:TYPE[?] RESolution NSRatio.....	6.17
SENSe:SWEEp:AVERAge:STATe[?] OFF ON.....	6.18
SENSe:SWEEp:AVERAge:TCONtrol[?] MOVing REPeat.....	6.18
SENSe:SWEEp:OFFSet:TIME[?] – (<TRIGger:DELay> + 0.005) to 100.0	6.18
SENSe:SWEEp:POINTs[?] 1 to 1024.....	6.18
SENSe:SWEEp:REALtime[?] OFF ON	6.18
SENSe:SWEEp:TIME[?] 0.0001 to 0.3.....	6.19
SENSe:TIMing:EXCLude:STARt[?] 0.0 to 0.1	6.19
SENSe:TIMing:EXCLude:STOP[?] 0.0 to 0.003.....	6.19
SYSTEM	6.21
SYSTem:INFO? [Item]	6.21
SYSTem:INITialize	6.22
SYSTem:MINPower?	6.22
SYSTem:TRANsaction:BEgin	6.23
SYSTem:TRANsaction:END	6.23
TEST	6.24
TEST:SENSor?	6.24
TRIGger	6.25
ABORt	6.25
INITiate:CONTInuous[?] OFF ON	6.25
INITiate:IMMediate	6.26
TRIGger:ATRigger:STATe[?] OFF ON	6.26
TRIGger:COUNT[?] 1 to 2×10^9	6.26
TRIGger:DELay[?] x to 100.0.....	6.26
TRIGger:DELay:AUTO[?] OFF ON.....	6.27
TRIGger:HOLDoff[?] 0.0 to 10.0	6.27
TRIGger:HYSteresis[?] 0.0 to 10.0	6.27
TRIGger:IMMediate.....	6.27
TRIGger:LEVel[?] x to y	6.27
TRIGger:SLOPe[?] POSitive NEGative	6.28
TRIGger:SOURce[?] BUS EXTernal HOLD IMMediate INTernal	6.28
List of Remote-Control Commands	6.29

Figs.

Fig. 6-1 Effect of SENSE:POWer:BURSt:DTOLerance..... 6.14
 Fig. 6-2 Effect of SENSE:TIMing:EXCLude:STARt and :END in the *Burst Average* mode..... 6.19
 Fig. 6-3 Effect of SENSE:TIMing:EXCLude:STARt and :END in the *Timeslot* mode..... 6.20

Tables

Table 6-1 Commands of the *CALibration* system..... 6.3
 Table 6-2 Commands of the *SENSe* system..... 6.5
 Table 6-3 Measurement modes..... 6.11
 Table 6-4 Optimum selection of the sampling window size (N = 1, 2, 3, ...)..... 6.12
 Table 6-5 Commands of the *SYSTem* system 6.21
 Table 6-6 Meaning of *Item* in the *SYSTem:INFO?* command..... 6.21
 Table 6-7 Commands of the *TEST* system 6.24
 Table 6-8 Commands of the *TRIGger* system..... 6.25
 Table 6-9 List of remote-control commands..... 6.29

6 Remote Control – Commands

Notation

In the following sections, all commands implemented in the sensor are first listed in a table according to command systems and are then described in detail. The notation is largely in line with the SCPI standard.

Command tables For a quick overview of available commands, the commands are listed in a table before they are described. These tables contain the following four columns:

Command:	Commands and their tree structure.
Parameters:	Possible parameters.
Unit:	The basic unit of the physical parameters (must not be sent with parameters).
Remarks:	Identification of all commands <ul style="list-style-type: none"> • that have no query form • that are available as query only

Indentations The various levels of the SCPI command hierarchy are shown in the table by indentations to the right. The lower the level, the greater the indentation to the right. It should be noted that the complete notation of the command includes the higher levels too.

Example:

SENSe:AVERage:COUNt is represented in the table as follows:

```
SENSe      first level
  :AVERage  second level
    :COUNt  third level
```

In the individual description, the command is shown in full length. An example of the command is given at the end of the description.

[?]
?

A question mark in square brackets at the end of a command indicates that this command can either be used as a setting command (without question mark) or as a query (with question mark). If the question mark is not in square brackets, the command is a query only.

Example:

SENSe:POWer:AVG:APERture[?]

SENSe:POWer:AVG:APERture 1e-3 sets the length of the sampling window to 1 ms.

SENSe:POWer:AVG:APERture? Returns the currently set length as a response.

**IDN?* Queries the sensor identification string that of course cannot be changed. For this reason, this command is only available as a query.

Special characters | for parameters

A vertical bar between parameters is used to separate alternative options (OR link).

Example:

NITiate:CONTInuous OFF | ON

The parameter *OFF* or *ON* can be entered.

{numeric expression}

A numeric expression in braces means that it has been rounded to the nearest integral value.

**<parameter>
<variable>**

A parameter or a variable in triangular brackets expresses its current value.

Commands as per IEEE 488.2

The sensor supports a subset of the possible setting commands and queries (*Common Commands and Queries*) in line with IEEE 488.2.

*IDN? – IDentification Query

*IDN? returns a string with information on the sensor's identity (device identification code). In addition, the version number of the installed firmware is indicated. The string for a sensor of type R&S NRP-Z21 has the following structure:

ROHDE&SCHWARZ,NRP-Z21,<serial number>,<firmware version>

<serial number>: Serial number in ASCII

<firmware version>: Firmware version number in ASCII

*RST – Reset

*RST sets the sensor to the default state, i.e. the default settings for all test parameters are loaded.

*TRG – Trigger

*TRG triggers a measurement. For this purpose, the sensor is in the *WAIT_FOR_TRIGGER* state and the source for the trigger event is set to *BUS (TRIGger:SOURce BUS)*.

*TST? – Self Test Query

*TST? starts a selftest and returns 0 (no error found) or 1 (an error has occurred). The selftest comprises the following functions:

- RAM test
- Operating voltages
- Temperature measurement
- Calibration data set
- Noise
- Zero-point offsets.

SCPI Commands

The sensors R&S NRP-Z11/-Z21 are controlled via the groups of commands

- CALibration (zeroing)
- SENSE (measurement configurations)
- SYSTem
- TRIGger
- SERvice.

CALibration

Table 6-1 Commands of the *CALibration* system

Command	Parameter	Unit	Remarks
CALibration			
:DATA[?]	<calibration data set as definite length block>		
:LENGth?		Bytes	Query only
:ZERO			
:AUTO[?]	OFF ON ONCE		

CALibration:DATA[?] <calibration data set as *definite length block*>

CALibration:DATA is used for writing a calibration data set in the flash memory of the sensor.

The query yields the calibration data set currently stored in the flash memory as a *definite length block*.

CALibration:DATA:LENGth?

CALibration:DATA:LENGth? yields the length in bytes of the calibration data set currently stored in the flash memory. Programs that read out the calibration data set can use this information to determine the capacity of the buffer memory required.

CALibration:ZERO:AUTO[?] OFF | ON | ONCE

The commands *CALibration:ZERO:AUTO ON* and *CALibration:ZERO:AUTO ONCE* zeroes the three measurement paths of the sensor. For this purpose, the test signal must be deactivated or the sensor disconnected from the signal source. The sensor automatically detects the presence of any significant power to be measured. This causes zeroing to be aborted and error message *NRPEROR_CALZERO* to be output. The *CALibration:ZERO:AUTO OFF* is ignored. Zeroing takes four seconds at a minimum, but at least as long as the selected averaging filter needs for settling (only fixed-filter mode).

*Repeat zeroing*

- *during warm-up after switching on or connecting the instrument*
- *after a substantial variation of the ambient temperature*
- *after fastening the sensor to an RF connector at high temperature*
- *after several hours of operation*
- *when very low-power signals are to be measured, e.g. less than 10 dB above the lower measurement limit.*

For zeroing switch off the test signal and do not remove the sensor from the signal source. Apart from keeping the thermal balance, this has the advantage that the noise superimposed on the test signal (e.g. from a broadband amplifier) can be detected on zeroing and does not impair the measurement result.

The query always yields 1 (= OFF).

Default setting

After a power-on reset, the zero offsets determined during the last calibration are used until the first zeroing. Therefore, very slight zero offsets are to be expected with a warmed up sensor. Initialization by means of **RST* or *SYSTem:INITialize* has no influence on the current zero offsets.

SENSe (Sensor Configuration)

The sensor is configured by means of the commands of the groups *SENSe* and *TRIGger*.

Table 6-2 Commands of the *SENSe* system

Command	Parameter	Unit	Remarks
SENSe			
:AVERage			
:STATe[?]	OFF ON		
:TCONtrol[?]	MOVing REPeat		
:COUNt[?]	1 to 65536		
:AUTO[?]	OFF ON ONCE		
:TYPE[?]	RESolution NSRatio		
:MTIME[?]	1.0 to 999.99	s	
:NSRatio[?]	0.0001 to 1.0	dB	
:RESolution[?]	1 to 4		
:SLOT[?]	1 to <SENSe:POWer:TSLot:AVG:COUNt>		
:CORRection			
:OFFSet[?]	-100.0 to 100.0	dB	
:STATe[?]	OFF to ON		
:DCYCLE[?]	0.001 to 99.999	%	
:STATe[?]	OFF to ON		
:SPDevice:STATe[?]	OFF to ON		
:FREQuency[?]	0.0 to 100.0e9	Hz	
:FUNction[?]	"POWer:AVG" "POWer:TSLot:AVG" "POWer:BURSt:AVG" "XTIME:POWer"		
:POWer			
:AVG			
:APERture[?]	1.0e-6 to 100.0e-3	s	
:BUFFer			
:STATe[?]	OFF ON		
:SIZE[?]	1 to 1024		

Command	Parameter	Unit	Remarks
:SMOothing:STATe[?]	OFF ON		
:BURSt:DTOLerance[?]	0.0 to 3.0e-3	s	
:TSLot:AVG			
:COUNt[?]	1 to 128		
:WIDTh[?]	10e-6 to 100e-3	s	
:RANGe [?]	0 to 2		
:AUTO[?]	OFF ON		
:CLEVel[?]	0.0 to 20.0	dB	
:SAMPling[?]	FREQ1 FREQ2		
:SGAMma			
:CORRection:STATe[?]	OFF ON		
:MAGNitude[?]	0.0 to 1.0		
:PHASe[?]	-360.0 to 360.0	degree	
:SWEep			
:AVERage			
:STATe[?]	OFF ON		
:TCONtrol[?]	MOVing REPeat		
:COUNt[?]	1 to 65536		
:AUTO[?]	OFF ON ONCE		
TYPE[?]	RESolution NSRatio		
:MTIME[?]	1.0 to 999.99	s	
:NSRatio[?]	0.0001 to 1.0	dB	
:RESolution[?]	1 to 4		
:POINt[?]	1 to <SENSe:SWEEp:POINts>		
:OFFSet:TIME[?]	- (<TRIGger:DELay> + 0.005) to 100.0	s	
:POINts[?]	1 to 1024		
:REALtime[?]	OFF ON		
:TIME[?]	0.0001 to 0.3		

Command	Parameter	Unit	Remarks
:TIMing			
:EXCLude			
:STARt[?]	0.0 to 3.0e-3	s	
:STOP[?]	0.0 to 3.0e-3	s	

SENSe:AVERage:COUNT[?] 1 to 65536

SENSe:AVERage:COUNT sets the number of measured values that have to be averaged for forming the measurement result in the modes *Continuous Average*, *Burst Average* and *Timeslot*. The higher this averaging factor, the less the measured values fluctuate and the longer the measurement time lasts. The parameter is rounded off to the nearest power-of-two number.

The query yields the averaging factor used in the modes *Continuous Average*, *Burst Average* and *Timeslot*.



The averaging function must be activated with SENSe:AVERage:STATe ON so that the set averaging factor becomes effective.

Default setting: 4

SENSe:AVERage:COUNT:AUTO[?] OFF | ON | ONCE

SENSe:AVERage:COUNT:AUTO activates (auto-averaging) or deactivates (fixed-filter mode) automatic determination of the averaging factor in the modes *Continuous Average*, *Burst Average* and *Timeslot*. If auto-averaging is activated, the averaging factor is continuously determined and set depending on the level of power and other parameters.

SENSe:AVERage:COUNT:AUTO ON activates auto-averaging and *SENSe:AVERage:COUNT:AUTO OFF* deactivates it. On deactivation, the previous, automatically determined averaging factor is used in the fixed-filter mode. The *SENSe:AVERage:COUNT:AUTO ONCE* command ensures that a new averaging factor is determined by the filter automatic function under the current measurement conditions and used in the fixed-filter mode.

The query yields

- 1 for *OFF*,
- 2 for *ON*.

Default setting: *OFF*

SENSe:AVERage:COUNT:AUTO:MTIME[?] 1.0 to 999.99

SENSe:AVERage:COUNT:AUTO:MTIME sets the settling time upper limit of the averaging filter in the auto-averaging mode for the *Continuous Average*, *Burst Average* and *Timeslot* modes and limits the length of the filter.

The query yields the current settling time upper limit of the averaging filter in the auto-averaging mode for the *Continuous Average*, *Burst Average* and *Timeslot* modes.

Default setting: 30.0 [s]

SENSe:AVERAge:COUNT:AUTO:NSRatio[?] 0.0001 to 1.0

SENSe:AVERAge:COUNT:AUTO:NSRatio determines the relative noise component in the measurement result for the *Continuous Average*, *Burst Average* and *Timeslot* modes if auto-averaging is operated in the corresponding mode (*SENSe:AVERAge:COUNT:AUTO:TYPE NSRatio*). The noise component is defined as the magnitude of the level variation in dB caused by the inherent noise of the sensor (two standard deviations).

The query yields the relative noise component in the result for the *Continuous Average*, *Burst Average* or *Timeslot* modes.

Default setting: 0.01 [dB]

SENSe:AVERAge:COUNT:AUTO:RESolution[?] 1 to 4

SENSe:AVERAge:COUNT:AUTO:RESolution sets the resolution index for the automatic averaging filter in the *Continuous Average*, *Burst Average* and *Timeslot* modes if it is operated in the *RESolution* mode. The resolution index equals the number of decimal places that have to be taken into account for the further processing of the measurement result in dBm, dB μ V or dB. The normal mode is designed in a similar manner as for the predecessors R&S NRVS and R&S NRVD or other commercial power meters. The higher the selected index, the better the measurement result is filtered without the last significant place (0.01 dB with an index of 3) actually being set. The *NSRatio* setting is recommended instead.

The query yields the resolution index for the *Continuous Average*, *Burst Average* and *Timeslot* modes.

Default setting: 3

SENSe:AVERAge:COUNT:AUTO:SLOT[?] 1 to <SENSe:POWer:TSLot:AVG:COUNT>

SENSe:AVERAge:COUNT:AUTO:SLOT defines the timeslot, whose power is referenced by auto-averaging in the *Timeslot* mode. The timeslot is addressed via its number, the counting beginning with 1. The timeslot number must not exceed the number of the currently set timeslots. If a valid timeslot number is initially set and then the number of timeslots reduced to a value that is smaller than the initial timeslot number, the initial value is automatically set to the new timeslot number, i.e. auto-averaging references the most recent timeslot.

The query yields the number of the current timeslot, whose power is referenced by auto-averaging in the *Timeslot* mode.

Default setting: 1

SENSe:AVERAge:COUNT:AUTO:TYPE[?] RESolution | NSRatio

SENSe:AVERAge:COUNT:AUTO:TYPE defines the automatic averaging filter mode in the *Continuous Average*, *Burst Average* and *Timeslot* modes. The *RESolution* parameter sets the mode usual for power meters; *NSRatio* predefines the compliance to an exactly defined noise component.

The query yields

- 1 for *RESolution*,
- 2 for *NSRatio*.

Default setting: *RESolution*

SENSe:AVERage:STATe[?] OFF | ON

SENSe:AVERage:STATe switches on or off the averaging filter for the *Continuous Average*, *Burst Average* and *Timeslot* modes.

The query yields

- 1 for *OFF*,
- 2 for *ON*.

Default setting: *ON*

SENSe:AVERage:TCONtrol[?] MOVing | REPeat

SENSe:AVERage:TCONtrol (*terminal control*) defines the behaviour of the averaging filter in the *Continuous Average*, *Burst Average* and *Timeslot* modes. As soon as a new measured value is shifted to the FIR filter, a new average value is available at the filter output, which is obtained from the new measured value and the other values stored in the filter.

The *MOVing* parameter defines that each new average value is output as a measurement result. This allows tendencies in the result to be recognized during the measurement procedure.

The *REPeat* parameter defines that a new result is output after the FIR filter has been filled with new measured values. This ensures that no redundant information is output.

The query yields

- 1 for *MOVing*,
- 2 for *REPeat*.

Default setting: *MOVing*

SENSe:CORRection:DCYClE[?] 0.001 to 99.999

SENSe:CORRection:DCYClE sets the duty cycle to a percent value for the correction of pulse-modulated signals. With the correction activated, the sensor calculates the signal pulse power from this value and the mean power. Since the duty cycle is only useful in the *Continuous Average* mode and the *normal mode*, it is evaluated only there.

The query yields the current duty cycle in percent.

Default setting: *1.0 [%]*

SENSe:CORRection:DCYClE:STATe[?] OFF | ON

SENSe:CORRection:DCYClE:STATe ON activates the duty cycle correction and thus the pulse-power measurement whereas *SENSe:CORRection:DCYClE:STATe OFF* deactivates it.

The query yields

- 1 for *OFF*,
- 2 for *ON*.

Default setting: *OFF*

SENSe:CORRection:OFFSet[?] -100.0 to 100.0

SENSe:CORRection:OFFSet defines a fixed offset in dB, which is used to correct the measured value. (When a log scale is used, the offset is added to the measured value; this is the reason why the command has this name.)

The attenuation of an attenuator located ahead of the sensor or the coupling attenuation of a directional coupler is taken into account with a positive offset, i.e. the sensor calculates the power at the input of the attenuator or directional coupler. A negative offset can be used to correct the influence of a gain connected ahead.

The query yields the set offset in dB.

Default setting: 0.0 [dB]

SENSe:CORRection:OFFSet:STATe[?] OFF | ON

SENSe:CORRection:OFFSet:STATe ON activates the offset correction and *SENSe:CORRection:OFFSet:STATe OFF* deactivates it.

The query yields

- 1 for *OFF*,
- 2 for *ON*.

Default setting: *OFF*

SENSe:CORRection:SPDev:STATe[?] OFF | ON

SENSe:CORRection:SPDevice:STATe ON activates the s-parameter data set for a component (attenuator, directional coupler) connected ahead of the sensor. Parameter *OFF* deactivates it.

The use of s-parameters instead of a fixed offset

(see group of commands *SENSe:CORRection:OFFSet*)

allows more precise measurements, since the interactions between the sensor, the source and components connected between them can be taken into account. (For detailed information on loading s-parameter data sets, refer to section 3.) The sensor has no factory-set s-parameter data set. In this state, the *SENSe:CORRection:SPDevice:STATe ON* command generates an error message.

The query yields

- 1 for *OFF*,
- 2 for *ON*.

Default setting:

The factory-set default setting of the sensor is *OFF*. On loading an s-parameter table, the default setting can be redefined (see section 3).

SENSe:FREQuency[?] 0.0 to 100.0e9

SENSe:FREQuency transfers the carrier frequency of the RF signal to be measured; this frequency is used for the frequency-response correction of the measurement result. The center frequency is set for broadband signals (*spread-spectrum* signals, multicarrier signals).

The query yields the set carrier frequency in Hz.

Default setting: 50.0e6 [Hz]

SENSe:FUNCTION[?] <sensor_function>

SENSe:FUNCTION <sensor_function> sets the sensor to one of the following measurement modes:

Table 6-3 Measurement modes

<sensor_function>	Description of the measurement mode
"POWer:AVG"	<p>Continuous Average After occurrence of the trigger event, the mean power is measured in a time interval (sampling window) whose width is defined with SENSE:POWer:AVG:APERTure. The single measurements are performed in pairs to obtain a more accurate measurement result by differentiation. With the averaging function activated, this operation is repeated the number of times specified by the averaging factor. With the averaging function activated, the actual measurement time is thus $2 \times \langle \text{SENSe:AVERage:COUNT} \rangle \times \langle \text{SENSe:POWer:AVG:APERTure} \rangle$ and with deactivated averaging function $2 \times \langle \text{SENSe:POWer:AVG:APERTure} \rangle$. Trigger events start one or several measurements in the Continuous Average mode (depending on the TRIGger:COUNT parameter).</p>
"POWer:TSLot:AVG"	<p>Timeslot The power is measured in a defined number of consecutive timeslots. Up to 128 timeslots have the same width, which is defined by SENSE:POWer:TSLot:WIDTH. Their quantity is defined by SENSE:POWer:TSLot:COUNT. The measurement result is an array, which contains as many elements as timeslots. Each element represents the average power in one of the timeslots. After each occurrence of the trigger event, a measurement is performed in all timeslots. The single measurements are also performed in pairs in this mode to obtain a more accurate measurement result by differentiation. With the averaging function activated, this operation is repeated the number of times specified by the averaging factor. With the averaging function activated, the actual measurement time is thus $2 \times \langle \text{SENSe:AVERage:COUNT} \rangle \times \langle \text{SENSe:POWer:TSLot:AVG:WIDTH} \rangle$ and with deactivated averaging function $2 \times \langle \text{SENSe:POWer:TSLot:AVG:WIDTH} \rangle$. Note that a corresponding number of trigger events must be available.</p>
"POWer:BURSt:AVG"	<p>Burst Average This mode is used for the measurement of the average power of repetitive single bursts, for example. The time interval (integration time) in which the average power is measured is not predefined. Instead, a special algorithm detects the beginning and end of the burst and ensures the synchronization of the sampling window. However, the user must define the dropout tolerance, a time criterion for detecting the pulse or burst end, using SENSE:POWer:BURSt:DTOLerance. In the Burst Average mode, external trigger events (irrespective of the setting of the TRIGger:SOURce parameter) and the TRIGger:DELay parameter are ignored.</p>
"XTIMe:POWer"	<p>Scope The Scope mode is similar to the Timeslot mode. Since, however, up to 1024 consecutive timeslots (called "points" here) are possible, the maximum averaging factor is limited to 8192 due to the storage capacity. In the Scope mode, other limits than in the Timeslot mode apply to the timeslot width. SENSE:SWEep:REALtime ON optimizes the time resolution. The differentiation and averaging filter are ignored.</p>

Time intervals that are excluded from the measurement can be set at the beginning and the end of the sampling window or timeslot in the measurement modes *Burst Average* and *Timeslot*. (*SENSe:TIMing:EXCLude:STARt* and *-:STOP*).

The query yields

- 1 for "POWer:AVG",
- 2 for "POWer:TSLot:AVG",
- 4 for "POWer:BURSt:AVG",
- 8 for "XTIMe:POWer".

Default setting: "POWer:AVG"

SENSe:POWer:AVG:APERture[?] 0.0001 to 0.1

SENSe:POWer:AVG:APERture defines the time interval (sampling window) for the *Continuous Average* mode; measured values are continuously recorded in this interval. In manual operation, the default setting of 20 ms in conjunction with the activated smoothing (see *SENSe:POWer:AVG:SMOothing:STATE*) is sufficient in most cases. Another value, which is normally higher, is required when the measurement result shows variations due to modulation. Especially with low-frequency modulation, it is useful to adapt the size of the sampling window exactly to the modulation period, which yields an optimum stable display.

Table 6-4 Optimum selection of the sampling window size (N = 1, 2, 3, ...)

Smoothing	Optimum sampling window size
OFF	$N \times \text{modulation period} / 2$
ON	$N \times \text{modulation period} \times 2$

The theoretically shortest measurement time can then be obtained only with smoothing deactivated. As the number of modulation periods that fit into a sampling window increases, the issue of whether N is an integer becomes more critical. With smoothing activated, approx. 5 periods are sufficient to reduce variations due to modulation to an acceptable extent; variations are no longer perceptible with more than 9 periods. With smoothing deactivated, the situation is significantly unfavourable. In this case, 5 instead of 300 periods are required and the variations completely disappear as of 3000 periods.

The query yields the currently set width of the sampling window in seconds.

Default setting: 0.02 [s]

SENSe:POWer:AVG:BUFFer:SIZE[?] 1 to 1024

SENSe:POWer:AVG:BUFFer:SIZE sets the buffer size for the buffered *Continuous Average* mode.

The query yields the current buffer size for the buffered *Continuous Average* mode.

Default setting: 1

SENSe:POWer:AVG:BUFFer:STATe[?] OFF | ON

The buffered *Continuous Average* mode is activated with *ON* and deactivated with *OFF*. In this mode, the results generated by trigger events are collected in the sensor until the buffer is filled. All results are then transferred as block data. The measurement rate obtained is thus higher than in the non-buffered *Continuous Average* mode. The maximum measurement rate is obtained by combining the buffered

mode with multiple triggering (see parameter *TRIGger:COUNT*). The size of the result buffer is set with the *SENSe:POWer:AVG:BUFFer:SIze* command.

The query yields

- 1 for *OFF*,
- 2 for *ON*.

Default setting: *OFF*

SENSe:POWer:AVG:SMOothing:STATe[?] OFF | ON

The *ON* parameter activates a smoothing filter for modulated signals in the *Continuous Average* mode and *OFF* deactivates it. The smoothing filter is a steep-edge digital lowpass filter used to suppress variations of results caused by low-frequency modulation. This parameter should be activated to reduce variations in results due to modulation when the size of the sampling window cannot or should not be exactly adapted to the modulation period. If the selected sampling window is 5 to 9 times larger than a modulation period, the variations in display are normally sufficiently reduced. With smoothing deactivated, 300 to 3000 periods are required to obtain the same effect.

With smoothing deactivated, the sampling values are considered equivalent and averaged in a sampling window, which yields an integrating behaviour of the measuring instrument. As described above, optimum suppression of variations in the result is thus obtained when the size of the sampling window is exactly adapted to the modulation period. Otherwise, the modulation can have a considerable influence, even if the sampling window is much larger than the modulation period. The behaviour can be considerably improved by subjecting sampling values to weighting (raised-von-Hann window), which corresponds to video filtering. This is exactly what happens with activated smoothing.

Since the smoothing filter increases the inherent noise of the sensor by approx. 20%, it should remain deactivated if it is not required.

The query yields

- 1 for *OFF*,
- 2 for *ON*.

Default setting: *ON*

SENSe:POWer:BURSt:DTOLerance[?] 0.0 to 0.003

SENSe:POWer:BURSt:DTOLerance defines the dropout tolerance, a parameter for reliably detecting the burst end in the *Burst Average* mode with modulated signals (e. g. with digital standards NADC, PDC, PHS, etc.). The dropout tolerance should be selected larger than the longest amplitude drop and smaller than the gap between two consecutive bursts. The default value is sufficient for all usual digital communication standards.

The query yields the dropout tolerance for the *Burst Average* mode.

Default setting: *0.0001 [s]*

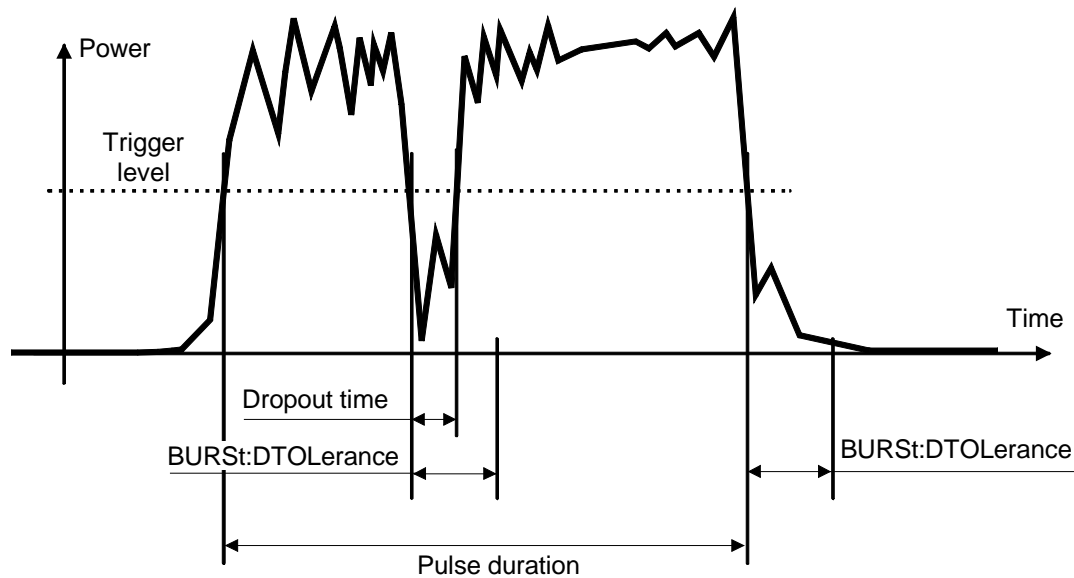


Fig. 6-1 Effect of `SENSe:POWer:BURSt:DTOLerance`

`SENSe:POWer:TSLot:AVG:COUNT[?]` 1 to 128

For the Timeslot mode, `SENSe:POWer:TSLot:AVG:COUNT` sets the number of consecutive timeslots that are to be processed after each trigger event.

The query yields the number of consecutive timeslots.

Default setting: 1

`SENSe:POWer:TSLot:AVG:WIDTH[?]` 0.0001 to 0.1

`SENSe:POWer:TSLot:AVG:WIDTH` sets the length of a timeslot in seconds for the Timeslot mode.

The query yields the length of a timeslot in seconds for the Timeslot mode.

Default setting: 0.001 [s]

`SENSe:RANGe[?]` 0 to 2

`SENSe:RANGe` selects the measurement path of the sensor. The sensor has three separate measurement paths. Path 1 is the most sensitive, path 2 medium and path 3 the least sensitive. `SENSe:RANGe 0` selects path 1, `SENSe:RANGe 1` path 2 and `SENSe:RANGe 2` path 3.

The dynamic ranges of these measurement paths depend on temperature and are model-specific. Reference values are 40 μ W (-14 dBm) for the most sensitive measurement path, 4 mW (6 dBm) for the medium one and 400 mW (26 dBm) for the least sensitive.

The query yields

- 0 for path 1,
- 1 for path 2,
- 2 for path 3.

If the measurement path is selected manually (`SENSe:RANGe:AUTO OFF`), the currently selected measurement path is output. With automatic selection, the last path that was set manually is output. It is therefore immediately reset after deactivating the automatic function.

Default setting: 2 (least sensitive path)

SENSe:RANGe:AUTO[?] OFF | ON

SENSe:RANGe:AUTO ON activates the automatic selection of the measurement path and *SENSe:RANGe:AUTO OFF* deactivates it.

The query yields

- 1 for *OFF*,
- 2 for *ON*.

Default setting: *ON*

SENSe:RANGe:AUTO:CLeVel[?] -20.0 to 0.0

SENSe:RANGe:AUTO:CLeVel is used to reduce the transition range between measurement paths 1 and 2 or 2 and 3 by the indicated value (in dB). This can improve the measurement accuracy for signals with a high *peak-to-average* ratio, since the headroom for modulation peaks becomes larger. The disadvantage is that the S/N ratio is reduced at the lower limits of the transition ranges.

The query yields the offset of transition ranges between measurement channels 1 and 2 or 2 and 3.

Default setting: *0.0 [dB]*

SENSe:SAMPling[?] FREQ1 | FREQ2

SENSe:SAMPling is used to vary the sampling frequency of the analog-digital converter in the sensor. With parameter specification *FREQ1* the sampling frequency is 133.358 kHz, and 119.467 kHz with parameter *FREQ2*. This is provided to suppress interfering low-frequency mixture products from signal components and the sampling frequency.

The query yields

- 1 for *FREQ1*,
- 2 for *FREQ2*.

Default setting: *FREQ1*

SENSe:SGAMma:CORRection:STATe[?] OFF | ON

SENSe:SGAMma:CORRection:STATe ON initiates the use of the complex reflection coefficient of the source defined with *SENSe:SGAMma:MAGNitude* and *SENSe:SGAMma:PHASe* for the correction of interactions between the sensor, the source and the components connected between them (see *SENSe:CORRection:SPDevice:STATe*). This compensates for the source mismatch, which often largely contributes to measurement uncertainty.

The query yields

- 1 for *OFF*,
- 2 for *ON*.

Default setting: *OFF*

SENSe:SGAMma:MAGNitude[?] 0.0 to 1.0

SENSe:SGAMma:MAGNitude defines the magnitude of the complex reflection coefficient of the source. A value of *0.0* corresponds to an ideal matched source and a value of *1.0* to total reflection.

The query yields the set magnitude.

Default setting: *0.0*

SENSe:SGAMma:PHASe[?] -360.0 to 360.0

SENSe:SGAMma:MAGNitude defines the phase angle (in degrees) of the complex reflection coefficient of the source.

The query yields the set phase angle.

Default setting: 0.0 [°]

SENSe:SWEep:AVERage:COUNT[?] 1 to 65536

SENSe:SWEep:AVERage:COUNT is used to set the number of measured values to be averaged in the *Scope* mode for the formation of the measurement result. The higher this averaging factor, the lesser the fluctuation the measured values and the longer the measurement time. The parameter is rounded off to the next power-of-two number.

The query yields the averaging factor used in the *Scope* mode.



The averaging function must be activated with SENSe:SWEep:AVERage:STATE ON so that the set averaging factor becomes effective.

Default setting: 4

SENSe:SWEep:AVERage:COUNT:AUTO[?] OFF | ON | ONCE

SENSe:SWEep:AVERage:COUNT:AUTO activates (auto-averaging) or deactivates (fixed-filter mode) the automatic determination of the averaging factor in the *Scope* mode. If auto-averaging is activated, the averaging factor is continuously determined and set depending on the level of power and other parameters.

SENSe:SWEep:AVERage:COUNT:AUTO ON activates auto-averaging and *SENSe:SWEep:AVERage:COUNT:AUTO OFF* deactivates it. When deactivation occurs, the previous, automatically determined averaging factor is used in the fixed-filter mode. The *SENSe:SWEep:AVERage:COUNT:AUTO ONCE* command ensures that a new averaging factor is determined by the filter automatic function under the current measurement conditions and that this factor is used in the fixed-filter mode.

The query yields

- 1 for *OFF*,
- 2 for *ON*.

Default setting: *OFF*

SENSe:SWEep:AVERage:COUNT:AUTO:MTIME[?] 1.0 to 999.99

SENSe:SWEep:AVERage:COUNT:AUTO:MTIME sets the settling time upper limit of the averaging filter in the auto-averaging mode for the *Scope* mode and limits the length of the filter.

The query yields the current settling time upper limit of the averaging filter in the auto-averaging mode for the *Scope* mode.

Default setting: 30.0 [s]

SENSe:SWEEp:AVERAge:COUNT:AUTO:NSRatio[?] 0.0001 to 1.0

SENSe:SWEEp:AVERAge:COUNT:AUTO:NSRatio determines the relative noise component in the measurement result for the *Scope* mode if auto-averaging is operated in the corresponding mode (*SENSe:SWEEp:AVERAge:COUNT:AUTO:TYPE NSRatio*). The noise component is defined as the magnitude of the level variation in dB caused by the inherent noise of the sensor (two standard deviations).

The query yields the relative noise component in the measurement result in the *Scope* mode.

Default setting: 0.01 [dB]

SENSe:SWEEp:AVERAge:COUNT:AUTO:RESolution[?] 1 to 4

SENSe:SWEEp:AVERAge:COUNT:AUTO:RESolution sets the resolution index for the automatic averaging filter in the *Scope* mode when it is operated in the *RESolution* mode. The resolution index is equal to the number of decimal places that have to be taken into account for the further processing of the measurement result in dBm, dB μ V or dB. Thus, the design of the normal mode is similar to that for the predecessors R&S NRVS and R&S NRVD or other commercial power meters. The higher the selected index, the better the measurement result is filtered without the last significant place (0.01 dB with an index of 3) being actually set. The *NSRatio* setting is recommended instead.

The query yields the resolution index for the *Scope* mode.

Default setting: 3

SENSe:SWEEp:AVERAge:COUNT:AUTO:POINT[?] 1 to <SENSe:SWEEp:POINTS>

SENSe:SWEEp:AVERAge:COUNT:AUTO:SLOT defines the "point" whose power is referenced by auto-averaging in the *Scope* mode. The "point" is addressed via its number, the counting beginning with 1. The "point" number must not exceed the number of the currently set "points". If a valid "point" number is initially set and then the number of "points" reduced to a value that is smaller than the initial "point" number, the initial value is automatically set to the new "point" number, i.e. auto-averaging references the most recent "point".

The query yields the number of the current "point" whose power is referenced by auto-averaging in the *Scope* mode.

Default setting: 1

SENSe:SWEEp:AVERAge:COUNT:AUTO:TYPE[?] RESolution | NSRatio

SENSe:SWEEp:AVERAge:COUNT:AUTO:TYPE defines the mode of the automatic averaging filter in the *Scope* mode. The *RESolution* parameter sets the mode usual for power meters; *NSRatio* predefines the compliance to an exactly defined noise component.

The query yields

- 1 for *RESolution*,
- 2 for *NSRatio*.

Default setting: *RESolution*

SENSe:SWEEp:AVERAge:STATe[?] OFF | ON

SENSe:SWEEp:AVERAge:STATe switches the averaging filter on or off for the *Scope* mode.

The query yields

- 1 for *OFF*,
- 2 for *ON*.

Default setting: *ON*

SENSe:SWEEp:AVERAge:TCONtrol[?] MOVing | REPeat

SENSe:SWEEp:AVERAge:TCONtrol (*terminal control*) defines the behaviour of the averaging filter in the *Scope* mode. As soon as a new measured value is shifted to the FIR filter, a new mean is available at the filter output, which is obtained from the new measured value and the other values stored in the filter.

The *MOVing* parameter defines that each new average value is output as a measurement result. This allows trends in the result to be recognized during the measurement procedure.

The *REPeat* parameter defines that a new result is output after the FIR filter has been filled with new measured values. This ensures that no redundant information is output.

The query yields

- 1 for *MOVing*,
- 2 for *REPeat*.

Default setting: *MOVing*

SENSe:SWEEp:OFFSet:TIME[?] – (<TRIGger:DELay> + 0.005) to 100.0

This command is used to shift the timeslot for measured-data acquisition along the time axis in the *Scope* mode without modifying the value of *TRIGger:DELay*. Positive values yield an additional delay and negative values a correspondingly earlier measured-data acquisition.

The query yields the set time in seconds.

Default setting: *0.0 [s]*

SENSe:SWEEp:POINts[?] 1 to 1024

This command defines the time resolution of the measurement result. Each "point" represents a time interval whose duration is obtained from the length of the timeslot (command *SENSe:SWEEp:TIME*) divided by the number of "points". The measurement result for a "point" equals the average power over the associated time interval.

The query yields the number of set "points".

Default setting: *100*

SENSe:SWEEp:REALtime[?] OFF | ON

SENSe:SWEEp:REALtime ON suppresses the measured-value acquisition in pairs so that single operations can be recorded in this setting. Since the averaging filter of the sensor is not used, *SENSe:AVERAge:STATe* is ignored, but is not affected.

The query yields

- 1 for OFF,
- 2 for ON.

Default setting: OFF

SENSe:SWEEp:TIME[?] 0.0001 to 0.3

SENSe:SWEEp:TIME sets the duration of the timeslot in the *Scope* mode. This timeslot is divided into a number of equal intervals, in which the average power is determined. The number of intervals equals the number of test points, which is set with the command SENSe:SWEEp:POINts.

The query yields the duration of the timeslot in the *Scope* mode (in s).

Default setting: 0.01 [s]

SENSe:TIMing:EXCLude:START[?] 0.0 to 0.1

SENSe:TIMing:EXCLude:START defines the exclusion time at the beginning of the measurement window in the *Burst Average* (Fig. 6-2) and *Timeslot* (Fig. 6-3) modes.

The query yields the exclusion time at the beginning of the measurement window.

Default setting: 0.0 [s]

SENSe:TIMing:EXCLude:STOP[?] 0.0 to 0.003

SENSe:TIMing:EXCLude:STOP defines the exclusion time at the end of the measurement window in the *Burst Average* (Fig. 6-2) and *Timeslot* (Fig. 6-3) modes.

The query yields the exclusion time at the end of the measurement window.

Default setting: 0.0 [s]

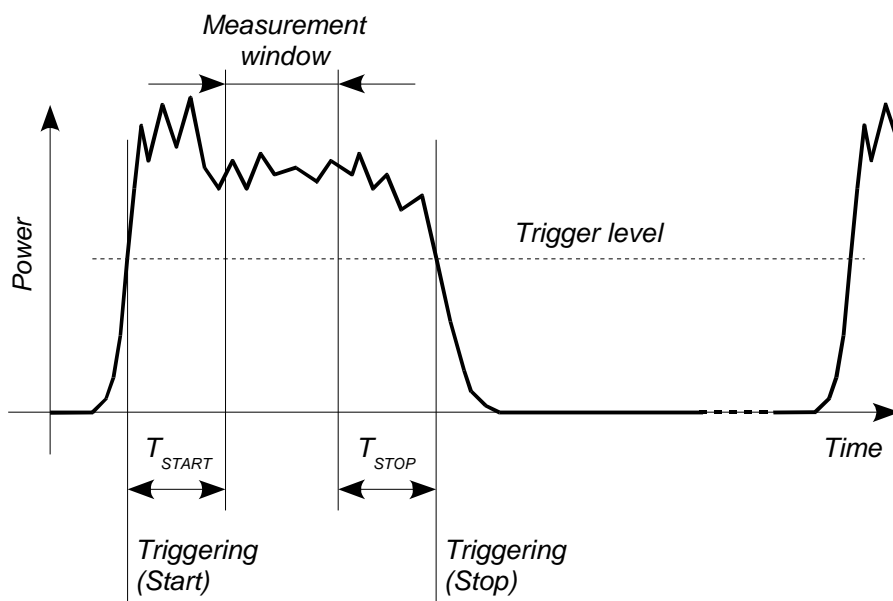


Fig. 6-2 Effect of SENSe:TIMing:EXCLude:START and :END in the *Burst Average* mode

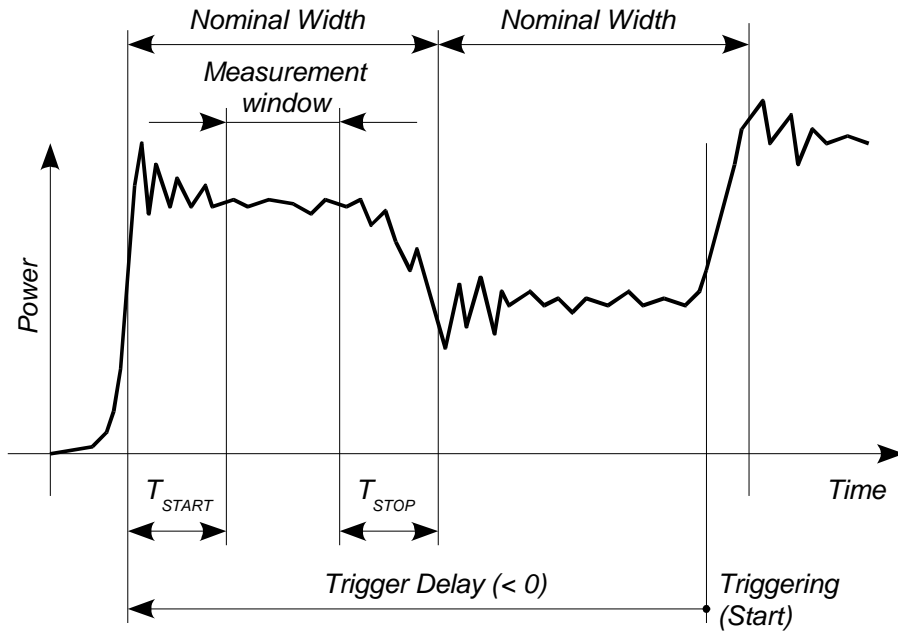


Fig. 6-3 Effect of `SENSe:TIMing:EXCLude:STARt` and `:END` in the *Timeslot* mode

SYSTEM

With the aid of the SYSTEM system, administrative device settings can be defined and queried. This includes detailed information on the sensor and its initialization and the transfer of available commands and their parameter limits.

Table 6-5 Commands of the SYSTEM system

Command	Parameter	Unit	Remarks
SYSTEM			
:INFO? [Item]			Query only
:INITialize			No query
MINPower?		W	Query only
:TRANsaction			
:BEGIN:			No query
:END			No query

SYSTEM:INFO? [Item]

SYSTEM:INFO? yields a string containing information that is more detailed than the identification string delivered by the sensor as a response to *IDN?. If no *Item* is specified, the response string is a sequence of entries in the form *Item:Information-String* separated by CR and LF (in C notation: \r\n). With the *Item* optionally appended to the command, the entry for the required *Item* can be queried. The response string is zero-terminated, i.e. its end identification is a zero byte (in C notation: \0).

Table 6-6 Meaning of *Item* in the SYSTEM:INFO? command

Item	Information string	Remarks
"MANUFACTURER"	"Rohde & Schwarz GmbH & Co. KG"	Manufacturer
"TYPE"	"NRP-Z11" or "NRP-Z21"	Type designation
"STOCK NUMBER"	"1138.3004.02" or "1137.6000.02"	Material number
"SERIAL"	"<serial number>"	6-digit serial number
"HWVERSION"	"000000000"	Hardware version (standard)
"HWVARIANT"	"000000000"	Hardware model (standard)
"SW BUILD"	"<build number>"	Version number of sensor firmware
"TECHNOLOGY"	"3 Path Diode"	Detector technology used
"FUNCTION"	"Power Terminating"	The R&S NRP-Z11/-Z21 is a termination power sensor.

Item	Information string	Remarks
"MINPOWER"	"<nominal lower test limit in W>"	The nominal lower test limit of the R&S NRP-Z11/-Z21 is 200 pW, i.e. with s-parameter correction deactivated, the sensor returns the information string "2e-10" as a response to "SYSTem:INFo? "MINPOWER". With s-parameter correction activated, the information string depends on the nominal lower limit of the sensor/twoport combination.
"MAXPOWER"	"<nominal upper test limit in W>"	The nominal upper test limit of the R&S NRP-Z11/-Z21 is 200 mW, i.e. with s-parameter correction deactivated, the sensor returns the information string "0.2" as a response to "SYSTem:INFo? "MAXPOWER". With s-parameter correction activated, the information string depends on the nominal upper limit of the sensor/twoport combination.
"IMPEDANCE"	"50"	The R&S NRP-Z11/-Z21 RF input has a nominal input impedance of 50 Ω .
"COUPLING"	"AC/DC"	The RF input of the R&S NRP-Z11/-Z21 is DC-coupled, but DC voltages superimposed on the RF signal are suppressed by the measurement amplifier.
"CAL. ABS."	"<date>"	Date of absolute calibration in the format YYYY-MM-DD. "Invalid Calibration Date" is returned with an invalid date entry.
"CAL. REFL."	"<date>"	Date of reflection-coefficient calibration in the format YYYY-MM-DD. "Invalid Calibration Date" is returned with an invalid date entry.
"CAL. S PARA."	"<date>"	Date of s-parameter calibration in the format YYYY-MM-DD. If no S parameter set is loaded, the sensor returns the string "not applicable". "Invalid Calibration Date" is returned with an invalid date entry.
"CAL. MISC."	"<date>"	Date of the calibration of other parameters in the format YYYY-MM-DD. "Invalid Calibration Date" is returned with an invalid date entry.
"SPD MNEMONIC"	"<mnemonic string>"	Clear-text designation of the components connected ahead of the sensor.

SYSTem:INITialize

SYSTem:INITialize sets the sensor to the standard state, i.e. the default settings for all test parameters are loaded in the same way as with **RST*. The sensor then outputs a complete list of all supported commands and parameters. With the command, the remote-control software can automatically adapt to the features of different types of sensors with different functionality.

SYSTem:MINPower?

SYSTem:MINPower? yields the lower test limit of the sensor or the combination comprising the sensor and components connected ahead of it, if the *SENSe:CORRection:SPDevice* parameter has the *ON* value. This query can be used to determine a useful resolution for the result display near the lower test limit.

SYSTEM:TRANSACTION:BEGIN

SYSTEM:TRANSACTION:BEGIN marks the beginning of a sequence of setting commands between which the parameter limits must not be checked. This prevents the display of error messages when a setting command causes a conflict that is resolved by a subsequent setting command. See *SYSTEM:TRANSACTION:END*.

SYSTEM:TRANSACTION:END

SYSTEM:TRANSACTION:END marks the end of a sequence of setting commands between which the parameter limits must not be checked. After this command, the parameter limits are checked.

TEST

Table 6-7 Commands of the TEST system

Command	Parameter	Unit	Remarks
TEST:SENSor?			Query only

TEST:SENSor?

TEST:SENSor? triggers a selftest of the sensor. In contrast to **TST*, this command yields detailed information, which is useful for troubleshooting.

TRIGger

Table 6-8 Commands of the TRIGger system

Command	Parameter	Unit	Remarks
ABORt			No query
INITiate			
:CONTinuous[?]	OFF ON		
:IMMediate			No query
TRIGger			
:ATRigger:STATe[?]	OFF ON		
:COUNt[?]	1 to 2×10^9		
:DELay[?]	x to 100.0	s	
:AUTO[?]	OFF ON		
:HOLDoff[?]	1.0e-9 to 10.0	s	
:HYSTeresis[?]	0.0 to 3.0	dB	
:IMMediate			No query
:LEVel[?]	x to y	W	
:SLOPe[?]	POSitive NEGative		
:SOURce[?]	BUS EXTernal HOLD IMMediate INTernal		

ABORt

ABORt interrupts the current measurement and sets the sensor to the *IDLE* state (normal case). However, if the sensor is in the continuous measurement mode (setting *INITiate:CONTinuous ON*), the *IDLE* state is immediately exited and the sensor enters the *WAIT_FOR_TRIGGER* state.

INITiate:CONTinuous[?] OFF | ON

INITiate:CONTinuous ON activates the continuous measurement mode. In this mode, a new measurement is automatically started when a measurement is terminated. The sensor first enters the *WAIT_FOR_TRIGGER* state and begins with the measurement as soon as the trigger condition is fulfilled. Once the measurement is completed, the sensor again enters the *WAIT_FOR_TRIGGER* state. The sensor will measure continuously assuming continuous trigger events.

In contrast, each measurement cycle must be explicitly started with the *INITiate:IMMediate* command after the *INITiate:CONTinuous OFF* command has been sent. After triggering and completion of the measurement, the sensor enters the *IDLE* status and remains in this status until a new measurement is started with the *INITiate:IMMediate* command.

The query yields

- 1 for OFF,
- 2 for ON.

Default setting: OFF

INITiate:IMMediate

INITiate:IMMediate starts a single measurement cycle. The sensor first changes from the *IDLE* state to the *WAIT_FOR_TRIGGER* state and begins with the measurement as soon as the trigger condition is fulfilled. Once the measurement is completed, the sensor again enters the *IDLE* state. Since the command is ignored during measurement, it normally has no effect in the continuous mode (setting *INITiate:CONTinuous ON*).

TRIGger:ATRigger:STATe[?] OFF | ON

TRIGger:ATRigger:STATe ON causes an artificial trigger event to be triggered if no trigger is recorded more than 10 s after the measurement has been started (only in the *Scope* mode). *TRIGger:ATRigger:STATe OFF* deactivates the trigger automatic function.

The query yields

- 1 for OFF,
- 2 for ON.

Default setting: OFF

TRIGger:COUNT[?] 1 to 2×10^9

This setting is designed for applications in which several consecutive measurements have to be performed by sending the *INITiate:IMMediate* command only once, e.g. to obtain a higher measurement speed. The gap between a single measurement and the continuous measurement mode is thus closed. The number of measurements is defined with the parameter associated with the *TRIGger:COUNT* command. This number equals the number of results yielded by the sensor at the end of the measurement.



The TRIGger:COUNT command does not define the number of trigger events required for performing the entire measurement task. The number may vary depending on the measurement mode.

*A further increase in the measurement speed can be obtained by combining the mode used with the buffered mode. The results are not made available immediately but as a block at the end of the measurement sequence (see group commands *SENSe:POWer:AVG:BUFFer*).*

The query yields the number of measurements performed with the *INIT:IMMediate* command after a measurement start.

Default setting: 1

TRIGger:DELay[?] x to 100.0

TRIGger:DELay defines the delay (in seconds) between the occurrence of the trigger event and the beginning of the measurement itself for the *Timeslot* and *Scope* modes. This parameter is ignored in the *Burst Average* mode. Pretriggering is obtained by means of negative values; with bus-triggered

measurements (see *TRIGger:SOURce*), the parameter must be set to positive values or zero to avoid measurement errors.

The query yields the set trigger delay for the *Timeslot* and *Scope* modes (in seconds).

Lower limit x of the parameter

Modes *Continuous Average*, *Burst Average* and *Timeslot*: $x = -0.005$

Mode *Scope*: $x = - (<SENSE:SWEEP:OFFSet:TIME> + 0.005)$

Default setting: 0.0 [s]

TRIGger:DElay:AUTO[?] OFF | ON

TRIGger:DElay:AUTO ON ensures by means of an automatically determined delay that a measurement is only started after the sensor has settled. This is important when thermal sensors are used. The automatically determined delay is ignored if a longer period was set with *TRIGger:DElay*. This does not overwrite the value of *TRIGger:DElay*. *TRIGger:DElay:AUTO OFF* deactivates this function.

The query yields

- 1 for OFF,
- 2 for ON.

Default setting: ON

TRIGger:HOLDoff[?] 0.0 to 10.0

TRIGger:HOLDoff suppresses trigger events within the set holdoff time (in s), starting from the time of the last successful triggering.

The query yields the set holdoff time (in s).

Default setting: 0.0 [s]

TRIGger:HYSteresis[?] 0.0 to 10.0

TRIGger:HYSteresis sets the hysteresis of the internal trigger threshold (parameter *TRIGger:LEVel*). Hysteresis is the magnitude (in dB) by which the trigger signal level falls below the trigger threshold (with positive trigger edge) to enable triggering again. The case is exactly the opposite with a negative trigger edge. The trigger hysteresis setting is only relevant to the *INTernal* trigger source.

The query yields the trigger hysteresis in dB.

Default setting: 0.0 [dB]

TRIGger:IMMediate

TRIGger:IMMediate triggers a generic trigger event that causes the sensor to exit immediately the *WAIT_FOR_TRIGGER* state irrespective of the trigger source and the trigger delay and begin with the measurement. The command is the only means of starting a measurement when the trigger source is set to *HOLD*.

TRIGger:LEVel[?] x to y

TRIGger:LEVel sets the trigger threshold for internal triggering derived from the test signal (in W). This setting is irrelevant to all other trigger sources.

The query yields the trigger threshold in W.

Lower limit x and upper limit y of parameter

SENSe:CORRection:OFFSet:STATe OFF: $x = 500 \times \langle \text{lower test limit} \rangle$
 $y = \langle \text{upper test limit} \rangle$

SENSe:CORRection:OFFSet:STATe ON: $x = 500 \times \langle \text{lower test limit} \rangle \times$
 $10^{\langle \text{SENSe:CORRection:OFFSet} \rangle / 10}$
 $y = \langle \text{upper test limit} \rangle \times$
 $10^{\langle \text{SENSe:CORRection:OFFSet} \rangle / 10}$

$\langle \text{lower test limit} \rangle:$ 200.0e-12 (with *SENSe:CORRection:SPDevice:STATe OFF*) or entered lower test limit of sensor/twoport combination (with *SENSe:CORRection:SPDevice:STATe ON*)

$\langle \text{upper test limit} \rangle:$ 0.2 (with *SENSe:CORRection:SPDevice:STATe OFF*) or entered upper test limit of sensor/twoport combination (with *SENSe:CORRection:SPDevice:STATe ON*)

Default setting: $10 \times x$

TRIGger:SLOPe[?] POSitive | NEGative

TRIGger:SLOPe defines the edge of the trigger event with internal and external triggering in the *Timeslot* and *Scope* modes. In this connection, positive means increasing envelope power (with internal triggering) or increasing voltage (with external triggering). As in the *Burst Average* mode, this command has no effect in conjunction with trigger sources *BUS*, *HOLD* and *IMMEDIATE*.

The query yields

- 1 for *POSitive*,
- 2 for *NEGative*.

Default setting: *POSitive*

TRIGger:SOURce[?] BUS | EXTernal | HOLD | IMMEDIATE | INTernal

TRIGger:SOURce sets the trigger source.

- *BUS:* Triggering with command **TRG* or *TRIGger:IMMEDIATE*.
- *EXTernal:* Triggering via USB Adapter R&S NRP-Z3. Relevant trigger parameters: *TRIGger:DELay* and *TRIGger:SLOPe*.
- *HOLD:* Triggering only with command *TRIGger:IMMEDIATE*.
- *IMMEDIATE:* Automatic triggering without explicit event.
- *INTernal:* Triggering by the measurement signal. Relevant trigger parameters: *TRIGger:LEVel*, *TRIGger:DELay* and *TRIGger:SLOPe* (not in the *Burst Average* mode).

The query yields

- 1 for *BUS*,
- 2 for *EXTernal*,
- 4 for *HOLD*,
- 8 for *IMMEDIATE*,
- 16 for *INTernal*.

Default setting: *IMMEDIATE*

List of Remote-Control Commands

The remote-control commands of the R&S NRP-Z11/-Z21 have a syntax based on standard SCPI 1999.0, but they comply with it only to a limited extent.

Table 6-9 List of remote-control commands

Command	Parameter	Unit	Default setting	Page
* Commands				
*IDN?				6.2
*RST				6.2
*TRG				6.2
*TST?				6.2
CALibration Commands				
CALibration:DATA[?]	<calibration data set as definite length block>			6.3
CALibration:DATA:LENGth?		Bytes		6.3
CALibration:ZERO:AUTO[?]	OFF ON ONCE		OFF (fixed)	6.4
SENSe Commands				
SENSe:AVERage:COUNT[?]	1 to 65536		4	6.7
SENSe:AVERage:COUNT:AUTO[?]	OFF ON ONCE		ON	6.7
SENSe:AVERage:COUNT:AUTO:MTIME[?]	1.0 to 999.99	s	30.0	6.7
SENSe:AVERage:COUNT:AUTO:NSRatIo[?]	0.0001 to 1.0	dB	0.01	6.8
SENSe:AVERage:COUNT:AUTO:RESolution[?]	1 to 4		3	6.8
SENSe:AVERage:COUNT:AUTO:SLOT[?]	1 to <SENSe:POWer:TSLot:AVG:COUNT>		1	6.8
SENSe:AVERage:COUNT:AUTO:TYPE[?]	RESolution NSRatIo		RESolution	6.8
SENSe:AVERage:STATe[?]	OFF ON		ON	6.9
SENSe:AVERage:TCONtroll[?]	MOVing REPeat		REPeat	6.9
SENSe:CORRection:DCYClE[?]	0.001 to 99.999	%	1.0	6.9
SENSe:CORRection:DCYClE:STATe[?]	OFF ON		OFF	6.9
SENSe:CORRection:OFFSet[?]	-100.0 to 100.0	dB	0.0	6.10
SENSe:CORRection:OFFSet:STATe[?]	OFF ON		OFF	6.10

Command	Parameter	Unit	Default setting	Page
SENSe:CORRection:SPDev:STATe[?]	OFF ON		OFF (can be modified by the user)	6.10
SENSe:FREQUency[?]	0.0 to 100.0e9	Hz	50.0e6	6.10
SENSe:FUNcTION[?]	"POWer:AVG" "POWer:TSLot:AVG" "POWer:BURSt:AVG" "XTIME:POWer"		"POWer:AVG"	6.11
SENSe:POWer:AVG:APERture[?]	0.0001 to 0.1	s	0.02	6.12
SENSe:POWer:AVG:BUFFer:SIZE[?]	1 to 1024		1	6.12
SENSe:POWer:AVG:BUFFer:STATe[?]	OFF ON		OFF	6.12
SENSe:POWer:AVG:SMOothing:STATe[?]	OFF ON		ON	6.13
SENSe:POWer:BURSt:DTOLerance[?]	0.0 to 0.003	s	0.0001	6.13
SENSe:POWer:TSLot:AVG:COUNt[?]	1 to 128		8	6.14
SENSe:POWer:TSLot:AVG:WIDTh[?]	0.0001 to 0.1	s	0.001	6.14
SENSe:RANGe[?]	0 to 2		2	6.14
SENSe:RANGe:AUTO[?]	OFF ON		ON	6.15
SENSe:RANGe:AUTO:CLEVel[?]	-20.0 to 0.0	dB	0.0	6.15
SENSe:SAMPLING[?]	FREQ1 FREQ2		FREQ1	6.15
SENSe:SGAMma:CORRection:STATe[?]	OFF ON		OFF	6.15
SENSe:SGAMma:MAGNitude[?]	0.0 to 1.0		0.0	6.15
SENSe:SGAMma:PHASe[?]	-360.0 to 360.0	degree	0.0	6.16
SENSe:SWEep:AVERage:COUNt[?]	1 to 65536		4	6.16
SENSe:SWEep:AVERage:COUNt:AUTO[?]	OFF ON ONCE		ON	6.16
SENSe:SWEep:AVERage:COUNt:AUTO:MTIME[?]	1.0 to 999.99	s	30.0	6.17
SENSe:SWEep:AVERage:COUNt:AUTO:NSRatio[?]	0.0001 to 1.0	dB	0.01	6.17
SENSe:SWEep:AVERage:COUNt:AUTO:POINt[?]	1 to <SENSe:SWEep:POINts>		1	6.17
SENSe:SWEep:AVERage:COUNt:AUTO:RESolution[?]	1 to 4		3	6.17
SENSe:SWEep:AVERage:COUNt:AUTO:TYPE[?]	RESolution NSRatio		RESolution	6.17
SENSe:SWEep:AVERage:STATe[?]	OFF ON		ON	6.18

Command	Parameter	Unit	Default setting	Page
SENSe:SWEEp:AVERAge:TCONtrol[?]	MOVing REPeat		REPeat	6.18
SENSe:SWEEp:OFFSet:TIME[?]	– (<TRIGger:DELay> + 0.005) to 100.0	s	0.0	6.18
SENSe:SWEEp:POINts[?]	1 to 1024		100	6.18
SENSe:SWEEp:REALtime[?]	OFF ON		OFF	6.18
SENSe:SWEEp:TIME[?]	0.0001 to 0.3	s	0.01	6.19
SENSe:TIMing:EXCLude:STARt[?]	0.0 to 0.003	s	0.0	6.19
SENSe:TIMing:EXCLude:STOP[?]	0.0 to 0.003	s	0.0	6.19
SYSTEM Commands				
SYSTEM:INFO? [Item]				6.21
SYSTEM:INITialize				6.22
SYSTEM:MINPower?		W		6.22
SYSTEM:TRANsaction:BEgIn				6.22
SYSTEM:TRANsaction:END				6.23
Test Commands				
TEST:SENSor?				6.24
Triggersystem Commands				
ABORt				6.25
INITiate:CONTinuous[?]	OFF ON		OFF	6.25
INITiate:IMMediate				6.26
TRIGger:ATRigger:STATe[?]	OFF ON		OFF	6.26
TRIGger:COUNt[?]	1 to 2×10 ⁹		1	6.26
TRIGger:DELay[?]	x to 100.0	s	0.0	6.26
TRIGger:DELay:AUTO[?]	OFF ON		OFF	6.27
TRIGger:HOLDoff[?]	1.0e-9 to 10.0	s	0.0	6.27
TRIGger:HYSTeresis[?]	0.0 to 10.0	dB	0.0	6.27
TRIGger:IMMediate				6.27
TRIGger:LEVel[?]	x to y	W	1.0e-6	6.28

Command	Parameter	Unit	Default setting	Page
TRIGger:SLOPe[?]	POSitive NEGative		POSitive	6.28
TRIGger:SOURce[?]	BUS EXTernal HOLD IMMEDIATE INTernal		IMMEDIATE	6.28
SERVICE Commands				
SERVice:CALibration:DITHer	ONCE		OFF	
SERVice:CALibration:DITHer:DATA?			0	
SERVice:CALibration:TEMP	ONCE		OFF	
SERVice:CALibration:TEMP:DATA?		K	0.0	
SERVice:CALibration:TEST[?]			-1	
SERVice:CALibration:ZERO:NEG0?			1	
SERVice:CALibration:ZERO:POS0?			2	
SERVice:CALibration:ZERO:NEG1?			3	
SERVice:CALibration:ZERO:POS1?			4	
SERVice:CALibration:ZERO:NEG2?			5	
SERVice:CALibration:ZERO:POS2?			6	
SERVice:DITHer	OFF ON		ON	
SERVice:MVCORrection[?]	0 to 63		63	
SERVice:PARAmeter:RTEMP[?]	<float value>	K	0.0	
SERVice:PARAmeter:RNULL1[?]	<float value>	Ω	0.0	
SERVice:PARAmeter:RNULL2[?]	<float value>	Ω	0.0	
SERVice:PARAmeter:RNULL3[?]	<float value>	Ω	0.0	
SERVice:PARAmeter:RBAHN[?]	<float value>	Ω	0.0	
SERVice:PARAmeter:NREF[?]	<float value>		0.0	
SERVice:PARAmeter:ATHERM[?]	<float value>	K^{-1}	0.0	
SERVice:PARAmeter:BTHERM[?]	<float value>	K	0.0	
SERVice:PARAmeter:CTHERM[?]	<float value>	K^{-1}	0.0	
SERVice:PARAmeter:DTHERM[?]	<float value>	K^{-1}	0.0	
SERVice:PARAmeter:CJUNC[?]	<float value>	F	0.0	
SERVice:RCCount[?]	1 to 32767		0	

Command	Parameter	Unit	Default setting	Page
SERVice:RESult[?]	0.0 to 1.0e6	W	0.0	
SERVice:SAMPle[?]	0 to 99999999		1000	
SERVice:UNLock	1234			

