Application Note

REAL-TIME RECEIVER TRACE EXPORT FOR R&S®ESW

Real-time capable SCPI export function for up to 4 traces simultaneously from the continuously running R&S®ESW equipped with R&S®ESW-B1000 wideband extension

Products:

► EMI Test Receiver R&S®ESW

► Wideband extension R&S®ESW-B1000/-B1000R and R&S®ESW-B350/-B350R

Tobias Groß | 1EP3 | Version 1.0 | 07.2024

https://www.rohde-schwarz.com/product/esw

ROHDE&SCHWARZ

Make ideas real

Contents

1 Overview

With the wideband extension R&S®ESW-B1000, the R&S®ESW EMI test receiver allows never seen testing speed with up to 970 MHz of FFT bandwidth. The receiver no longer forms the bottleneck for efficient EMI testing. The wideband extension allows to capture the full CISPR band C and D, from 30 MHz to 1 GHz, in a single shot. With unprecedented computing power, the option offers fully real-time measurements without any gaps in time between consecutive measurements.

External applications can now take advantage of the wide bandwidth. ESW firmware 3.20 adds a new method for exporting trace data from the instrument, regardless if the instrument is equipped with the wideband extension or not. SCPI commands allow live trace export of multiple traces simultaneously.

This application note describes the use and applications of the described SCPI commands to request trace data of up to 4 traces simultaneously from the R&S®ESW receiver, even allowing CISPR detectors. The commands enable trace export while the instrument measures continuously and therefore theoretically enable infinite measurements.

Example scripts, written in Python, demonstrate applications for capturing data from the instrument. The script detects if the real-time condition is violated. At short measurement times, the network or computation capabilities might not be able to cope with the large amount of data produced by the instrument. For wider frequency ranges that require multiple scan ranges, the real-time check can be disabled to allow combined FFT segments.

Figure 1 shows three scan traces (peak, quasi-peak and CISPR-average) in a Python graph window, captured from instrument and live displayed. With an adequate measurement time (or time interval per frame, with each frame consisting of one or up to 4 traces) this display works in real-time. If the real-time condition is not met, the script will stop and show an error.

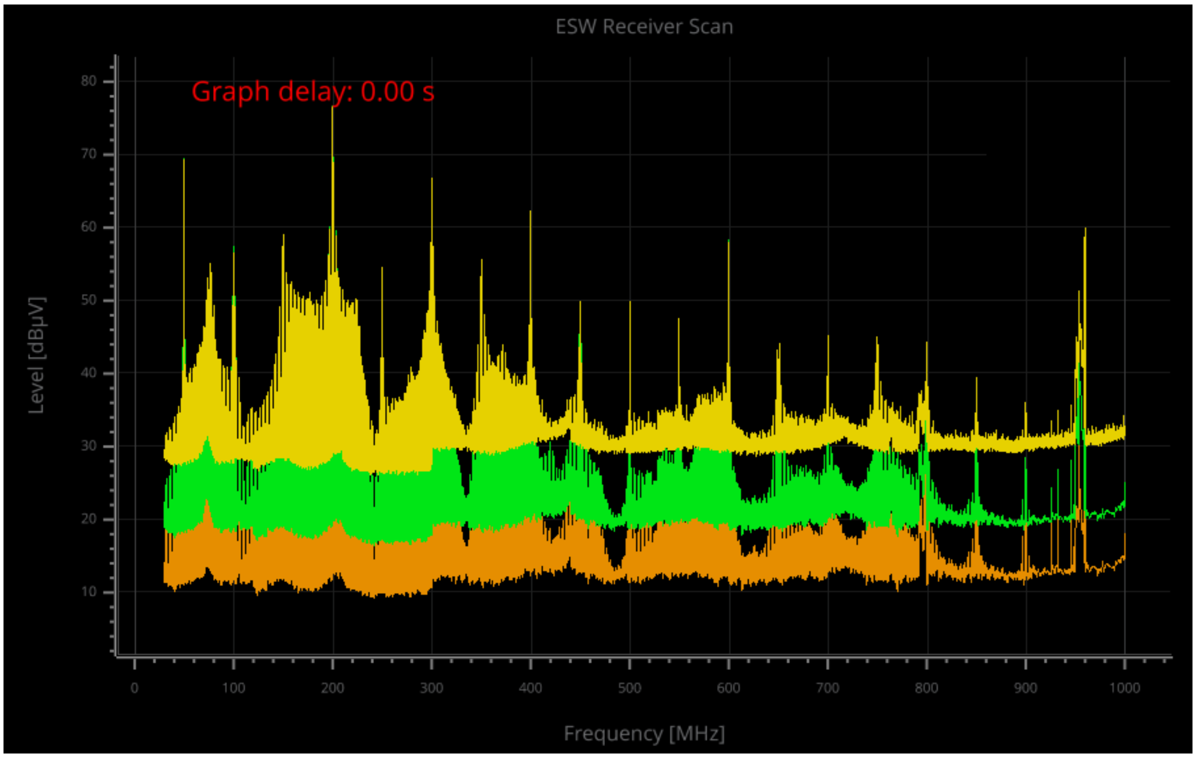

Figure 1 Python script exporting trace data from the R&S®ESW and simultaneously displaying them in real-time. In case the computer is not capable of processing the trace data fast enough, a delay between capturing on the instrument and displaying them on the computer will arise, denoting the delay on the graph. The measurement shows a comb generator, creating a wide bandwidth spectrum, measured with peak, quasi-peak and CISPR-average detector.

2 SCPI Programming Reference

The SCPI commands given in this section are used to request trace data while the receiver is running. At first, the *Multimode* needs to be switched on by setting CALC:SPEC:MMOD 1. Is recommended to verify it's status as units with firmware < 3.20 do not support this mode. Following, the availability of frames is checked using the command TRAC:SPEC:FINF?. After the first frames are available, the result will be e.g. 1,15 with 15 frames being available on the ring buffer (see section 2.2), starting from frame 1. TRAC:SPEC:FDAT 1,15 will then request those 15 first frames as a byte stream with the structure given in the following section.

2.1 SCPI command reference

Switching into Spectrogram Multimode

CALCulate:SPECtrogram:MMODe <Multimode>

Multimode is required to enable multiple (hidden) spectrograms for every detector to run in parallel. The spectrograms are not available on the instrument itself and not visible on the display and only used for the purpose of remote data capturing.

Parameter:

<Multimode>	ON | OFF | 1 | 0	*RST: 0

Example:

```
CALC:SPEC:MMOD 1
```

Requesting status of Spectrogram Multimode

CALCulate:SPECtrogram:MMODe?

It is recommended to verify the availability of Multimode to ensure the instruments firmware supports Spectrogram data export.

Return value:

<Multimode>	1 | 0

Example:

```
CALC:SPEC:MMOD?
//would return, e.g. 1
```

Requesting current available frames during Spectrogram measurement

TRACe[:DATA]:SPECtrogram:FINFo?

Return values:

<oldest>,<latest>

<oldest>	index of the oldest (first triggered) frame which is available (numeric value)
<latest>	index of the latest (last triggered) frame which is available (numeric value)

Return value of "-1,-1" means no frame(s) are available.

Example:

```
TRAC:SPEC:FINF?     //would return, e.g. 1,86
```

Requesting data of specific frames during Spectrogram measurement

TRACe[:DATA]:SPECtrogram:FDATa? <first>,<last>

Parameters:

<first>	index of the first frame to read
<last>	index of the last frame to read

Return value:
All frames with index <first> ≤ frame index ≤ <last> are sent in a byte stream. The prefix contains the length of bytes Y after the prefix in the ASCII coded prefix segment of maximum 10 bytes, starting with '#'. The byte stream ends with first and last available frame, equal to what TRACe[:DATA]:SPECtrogram:FINFo? returns.

Prefix (ASCII coded):

Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Value	#	X	Y1	Y2 (opt)	Y3 (opt)	Y4 (opt)	Y5 (opt)	Y6 (opt)	Y7 (opt)	Y8 (opt)	Y9 (opt)

X = Length of Y in bytes (range 1 ... 9)
Y = Total length of following binary data of all requested frames data in bytes

Binary Data

Number of frames			4 bytes (uint32)
Frame start timestamp (seconds since 01.01.1970)			8 bytes (double)
Frame start timestamp fraction in ns ($10^{-9}\ s$)			8 bytes (double)
Reduction Factor			4 bytes (uint32)
Loop 1 ≤ Frame Index ≤ Number of frames:			
	Frame Index		4 bytes (uint32)
	Number of traces		4 bytes (uint32)
	Loop 1 ≤ Trace Index ≤ Number of traces:		
		Trace Index	4 bytes (uint32)
		Status (bit 0 = overload flag)	1 byte (uint8)
		Trace stop timestamp seconds (seconds since 01.01.1970)	8 bytes (double)
		Trace stop timestamp fraction in ns ($10^{-9}\ s$)	8 bytes (double)
		Number of trace points	4 bytes (uint32)
		n data bytes	n bytes (float32) with $n = 4 \cdot$ trace points
Index of current oldest (first triggered) available frame			4 bytes (uint32)
Index of current latest (last triggered) available frame			4 bytes (uint32)

1 ≤ Number of traces ≤ 4

Rules of handling:

1. Error message `ERROR_INDEX_OUTOFRANGE` will be sent if:
   - The number of the last requested frame is lower than the number of the first
   - There is no overlap between available and requested range of frames
2. If any requested frame is no longer available in the buffer, the field “Number of traces” of the concerning frame will be set to 0 and no data bytes will follow.
3. If the next requested bar isn’t available yet, no further data will be sent for this request.

2.2 Frame ring buffer

The R&S®ESW stores frames in a ring buffer as shown in Figure 2. A new frame becomes available in the buffer as soon as it is fully captured (full span) and calculated. The first frame will start at index 1 with *n* counting upwards until the maximum number of frames m in the buffer is achieved. New incoming frames will then overwrite the oldest frames. Depending on the chosen measurement settings, the ring buffer will fill up quickly, allowing only a short interval to export these frames. If the exporting does not keep up with the generation of new frames by the instrument, the measurement cannot be considered as real-time in a system consideration. The external software needs to verify this condition by the frame time stamps.
The number of frames in the ring buffer *m* can be calculated from the maximum number of 10 Mio. trace point buffer in the R&S®ESW. It follows:

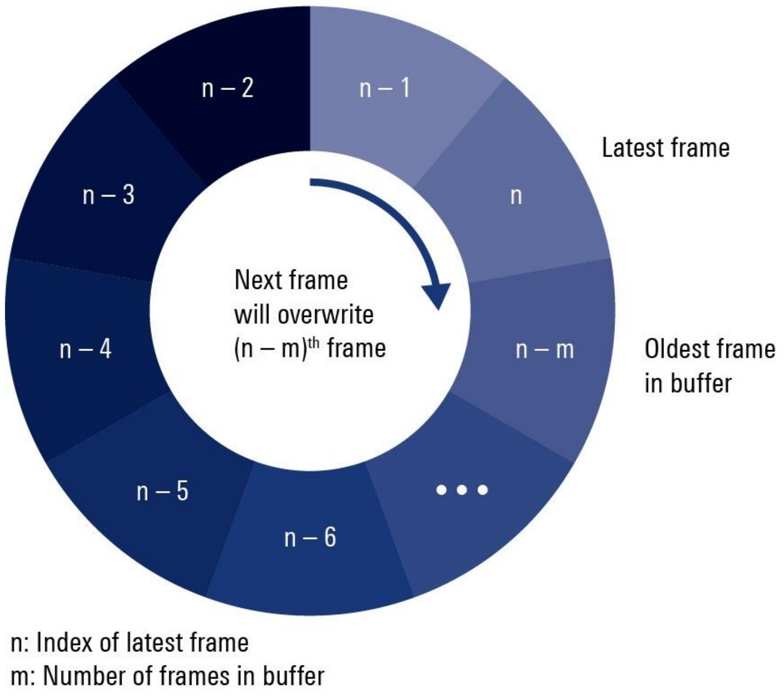

Figure 2 Frame ring buffer storing the latest m frames in the R&S®ESW.

$$m = \frac{10\ Mio.}{\frac{f_{Stop} - f_{Start}}{RBW/2} \cdot n_{Traces}}$$

3 Example Script Python

Unzip the files provided with this application note for the Python example program. All packages required to run the program are given in the "requirements.txt" file. These packages need to be installed before the script can be executed. The script consists of the following classes:

► **ESWRealtimeExport.py**

This is the main class that contains the __main__ function to execute the script. It contains the Instrument class that handles the instrument. Make sure to enter the right IP address of the ESW receiver in the __init__ of Instrument class.

For displaying the trace data, either ScanDisplay or SpectrogramDisplay can be used. In __main__ function, the corresponding class needs to be selected by either writing
ESWGrapher = ScanDisplay(ESWdata)
or
ESWGrapher = SpectrogramDisplay(ESWdata).

► **SpectrogramDisplay.py**

This class displays a spectrogram view of the trace data. The Update function is called in a loop ensuring the plot is always up to date. If data arrives to often and/or the computer is to slow to plot every trace, a plot delay will accumulate. However, all frames will be shown, even with a delay. The level color representation is fitted to the selected range with *settings.Ref* and *settings.Range*, given in *SpectrogramData* class.

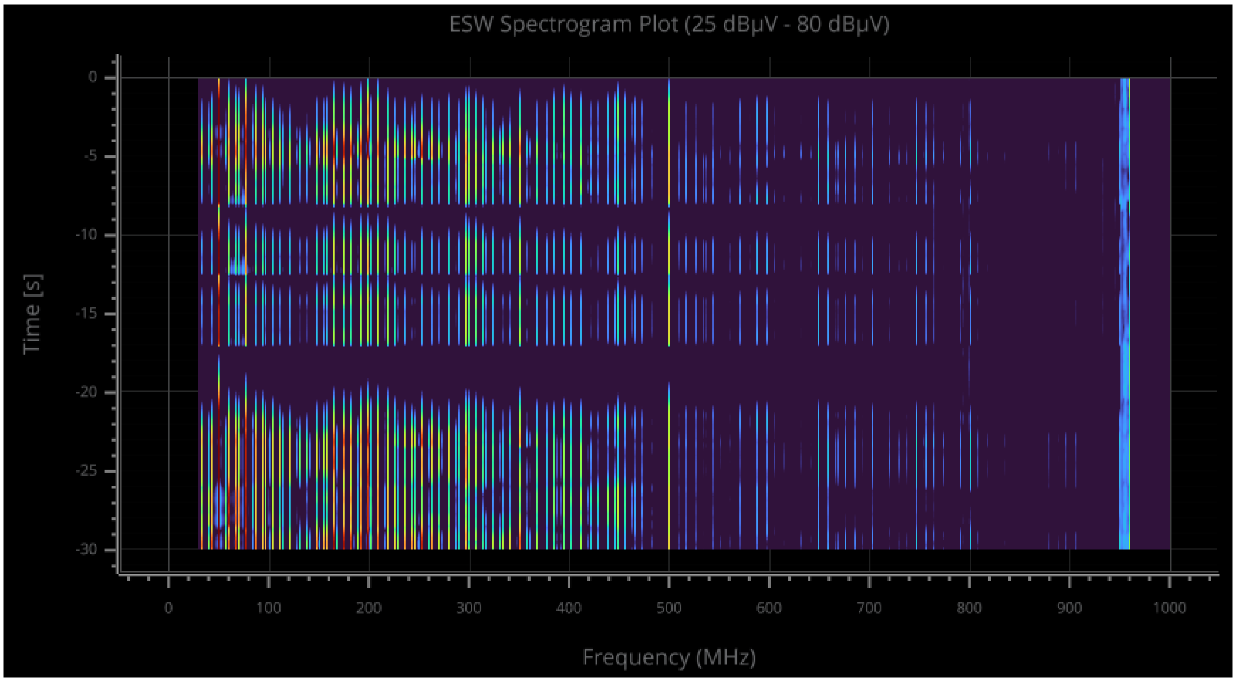

Figure 3 Spectrogram in free run mode to display real-time trace data.

► **ScanDisplay.py**

This class displays trace data in a scan display using Python package Vispy. The class contains an update function that is called in a loop ensuring the plot is always up to date. If data arrives to often and/or the computer is to slow to plot every frame, a plot delay will accumulate. This is indicated by the "Graph delay" message in the top left of the plot window. To ensure real time capability, all frames will be shown in the plot, even if they are plotted with a delay.

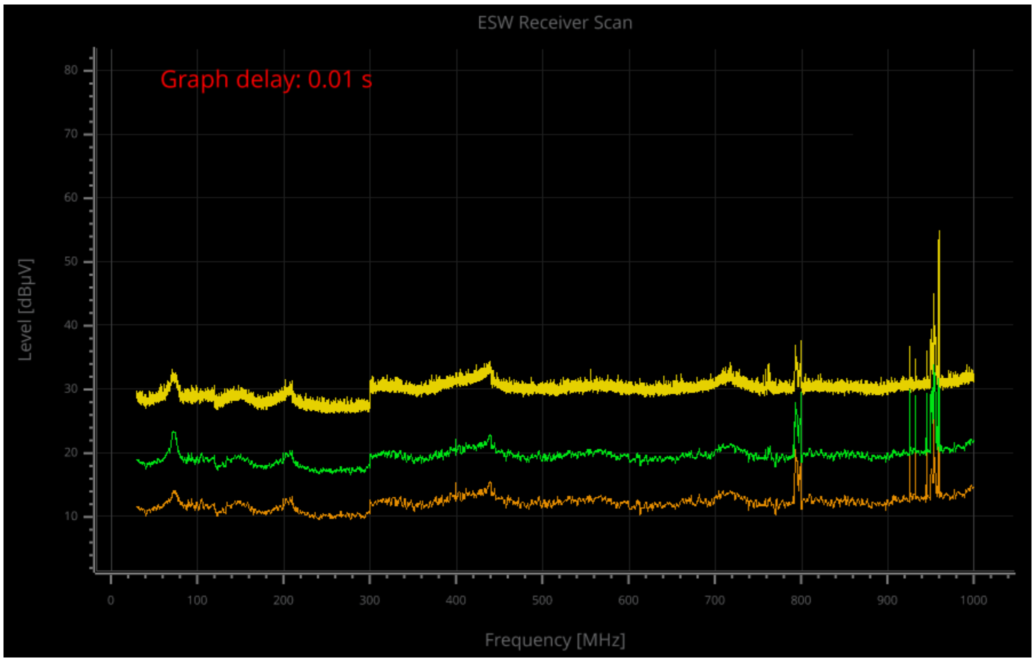

Figure 4 *ScanDisplay* showing three traces and their current delay between capturing and displaying.

In case a trigger is selected in the settings of *SpectrogramData* class, no trace is shown until the trigger is provoked. This is indicated by the message "Waiting for Trigger".

Figure 5 Waiting for the trigger to provoke.

► **RSSpectrogram.py**

This class performs the decoding of the received byte stream from the instrument. The structure of the byte stream that this class decomposes is given in section 2.1. *RSSpectrogram* class also performs mechanisms to ensure the transmission is still real time capable. For every frame received from the instrument, the timestamp is checked for consistency. At too low measurement time or a lack of computational or network resources, the transmission might not be real-time any more. In this case, a message is shown and the transmission is interrupted.

► SpectrogramData.py

This class handles the captured frames and their traces. The configuration of all receiver settings as start and stop frequency, measurement time, detectors, etc. is done in this class. This class also enables to save exported data to the disk, using binary or CSV format. Note that CSV export not only requires significantly more storage space, but also is the storing process much slower than saving to a binary file.

Receiver and measurement settings in SpectrogramData.py	
settings.Start	Start frequency in MHz
settings.Stop	Stop frequency in MHz
settings.Time	Measurement time in seconds
settings.Rbw	Resolution Bandwidth (RBW) in MHz
settings.Ref	Reference level in dBuV (upper limit on the level axis)
settings.Range	Display range in dB (calculated from reference level)
settings.Mode	Operating mode "Receiver" or "RTSpectrogram". Default is "Receiver".
settings.CheckRealTime	This enables the real-time check of incoming frames in RSSpectrogram.py
detectors	Detectors used for the measurement. Choose up to four of the following: AVER Selects the Average detector CAV Selects the CISPR Average detector CRMS Selects the CISPR RMS detector POS Selects the Max Peak detector QPE Selects the quasi-peak detector RMS Selects the RMS detector
export	Enables the exporting of trace data to disk in binary or CSV format
exportSubsampling	Saves every n-th element to the export file. A number greater than 1 will subsample trace data before saving. Default: 1
exportType	Binary or CSV file format. Binary format requires significantly less storage and saves faster, enabling faster measurements than saved with CSV format.
triggerEnable	Enables the trigger for exporting to file. Triggering is performed on the computer, not the receiver. Trigger invoking only influences when traces are saved to disk to avoid storage overflow.
triggerLimit	Threshold in dBuV for invoke the example trigger (see method trigger) for the implementation.
triggerDuration	Duration in seconds for which traces are saved to disk after trigger invoked.
Graph_size	Size of the graph to display. Default is 1500 by 800 pixels.

3.1 Long time measurement with triggering

Sporadic emissions lead to challenging testing with high requirements on the receiver to detect shortest pulses. EMI engineers might observe the DUT for much longer times than just seconds or minutes, even expanding to hours or days, permanently capturing the emission and displaying it in the spectrogram to get detailed understanding of their DUT's sporadic emissions. The instruments internal memory limits the duration for which the spectrogram can be stored internally on the instrument. After the maximum number of traces is achieved, the ring buffer starts to overwrite previously measured traces (see section 2.2).

With the trace export function, potential infinite measurements are possible. This script permanently captures data from the continuously running instrument. A triggering function allows to display and save only the relevant moments when sporadic emissions occur.

Here a simple threshold trigger is implemented that checks every trace point (across all selected detectors) for exceeding the chosen threshold value (*triggerLimit*). In such event, trace data is displayed (either on *ScanDisplay* or *SpectrogramDisplay*) and exported to a file for a selected period of time (*triggerDuration*). With the export to a file function turned on, trace data is only saved to the file during an active trigger event.

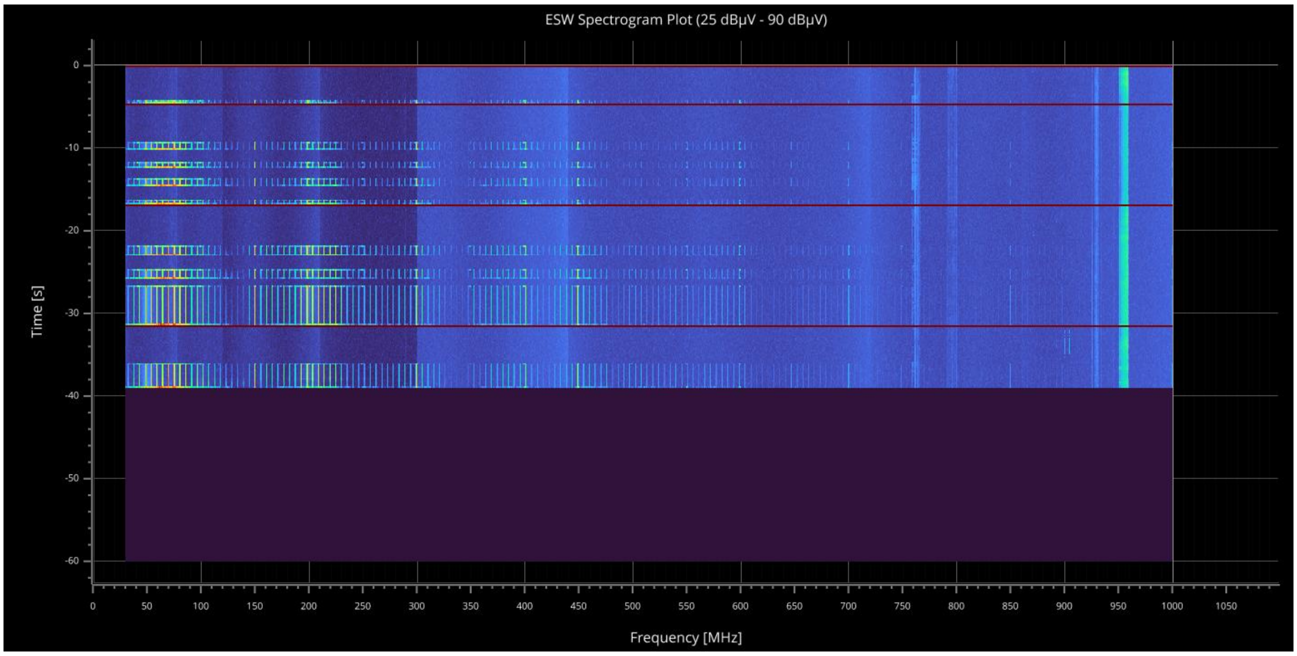

Figure 6 Spectrogram of a long-time measurement. Red lines in the spectrogram indicate the end of a trigger event. Trace data is shown until a defined time has passed since the last trigger provoked. Note that the time axis is only relative and can not be seen absolute with large gaps in between individual trigger events.

Rohde & Schwarz

The Rohde & Schwarz electronics group offers innovative solutions in the following business fields: test and measurement, broadcast and media, secure communications, cybersecurity, monitoring and network testing. Founded more than 80 years ago, the independent company which is headquartered in Munich, Germany, has an extensive sales and service network with locations in more than 70 countries.

www.rohde-schwarz.com

Certified Quality Management
ISO 9001

Rohde & Schwarz training

www.rohde-schwarz.com/training

Rohde & Schwarz customer support

www.rohde-schwarz.com/support