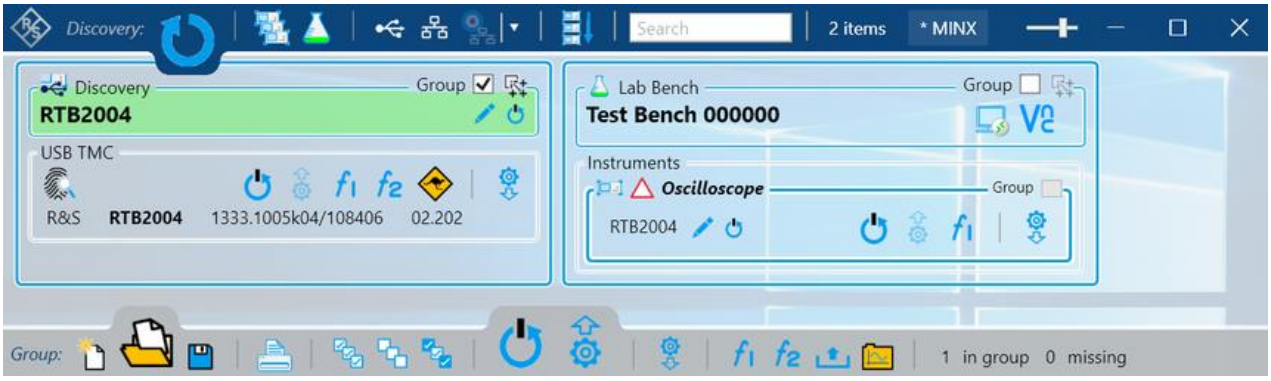
Application Note

# MINX USER-DEFINED INSTRUMENT FUNCTIONS

based on SCPI command sequences

**Products:**

- ► MILS-MINX
- ► MILS-MINION




Christian Wicke | GFM344 | Version 0e | 06.2020

https://www.rohde-schwarz.com/appnote/GFM344

# Contents


1 Overview ..... 3
2 Getting Started ..... 4
2.1 Prerequisites ..... 4
2.2 MINX Files and Folders ..... 5
3 ResourceCustomisationData.config Structure ..... 6
3.1 Element 'IsEdited' ..... 6
3.2 Element 'UserFunctions' ..... 6
3.3 Element 'SaveRecallCommands' ..... 9
3.4 Element 'UploadFunctions' ..... 11
3.5 Element 'GenericNames' ..... 12
4 References ..... 14
5 Ordering Information ..... 14
6 Appendix ..... 15
6.1 SCPI Session Window ..... 15

# 1 Overview

**MINX** – the **M**easurement **I**nstrument **N**etwork e**X**plorer – is a fast, non-invasive discovery and configuration software from Rohde & Schwarz for LAN and USB T&M instruments.

It's meant for application engineers, research and universities, providing an intuitive graphical user interface with access to powerful configuration, control, results collection and remote display functions.

Key benefits are

- ► SCPI based remote control.
- ► User function keys for screenshots, results download, etc.
- ► Save and recall of instrument settings.
- ► File upload functionality of arb and waveform files, including firmware updates.
- ► Copy settings between instruments or benches.
- ► Automatic grouping of instruments into 'virtual benches', with MINION.
- ► Status and report generation.
- ► Presentation and access of bench/instrument content via HTTP, RDC, VNC and FTP.
- ► Instrument vendor independent.

Moreover, MINX provides extended remote functionality making even more sophisticated instrument operations available by just the click of a single button.

Such extended functions are SCPI-based command sequences that are not standard for all instruments, which can be added on a per-instrument basis. Many of these extended functions already exist or can be added on request. This does not require a MINX software upgrade at all, just a restart of the software.

The following chapters of this application note are describing the different kind of extended function groups: **UserFunctions**, **SaveRecallFunctions** and **UploadFunctions**. Moreover, it will be pointed out how extended functions can be added or edited and which format the code must comply in order to get available in MINX.

For further details about the installation and usage of MINX, please refer to the MINX user manual.

# 2 Getting Started

## 2.1 Prerequisites

### Mandatory

- RSUK MINX (version 4.0 or higher) [1]

### Optional but recommended

- Notepad++ [2]
- XML Notepad 2007 [3]

### Recommendations

- For using Notepad++ make sure to enable syntax highlighting for XML. Invalid (e.g. badly formed or broken) XML will cause MINX to reset and overwrite `ResourceCustomisationData.config` with a default template on startup.

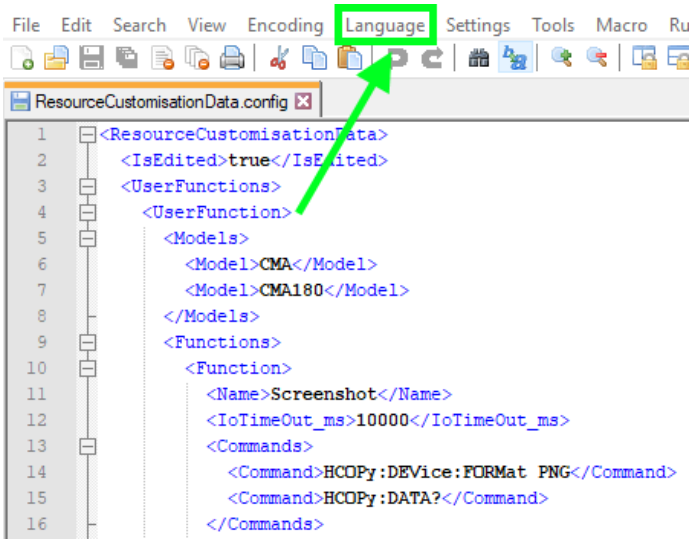



Figure 1: Displaying ResourceCustomisationData.config with Notepad++ and XML syntax highlighting

- In all cases, before making changes and modifications to `ResourceCustomisationData.config`, do make a backup copy.
- In order to make sure the SCPI command sequence is working properly and resulting the expected behavior, check it with the **MINX SCPI Session** terminal (6.1) in advance. There, SCPI commands can be sent individually or scripted (without flow control). This is the recommended way to debug SCPI sequences, as there is no debugging in the config file and SCPI errors may hang the instrument.

## 2.2 MINX Files and Folders

The software is installed with the following default settings:

| Type | File Path |
|---|---|
| **Installation directory** | `C:\Program Files (x86)\RSUK MINX` |
| **MINX user manual** | `<InstallDir>\MILS User Manual.pdf` |
| **User directory** | `C:\Users\<UserDir>\Documents\My MINX` |
| **User functions results directory (e.g. device screenshots)** | `C:\Users\<UserDir>\Documents\My MINX\User Function Results` |
| **Resource configuration file (for defining user-specific instruments functions)** | `C:\Users\<UserDir>\Documents\My MINX\Configuration\ResourceCustomisationData.config` |

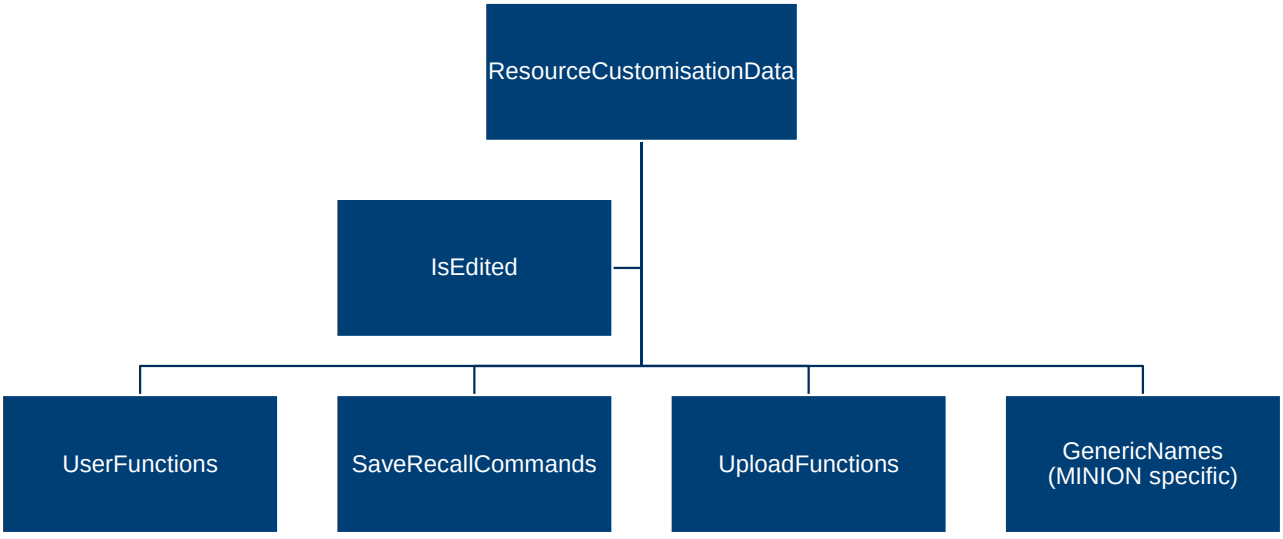
The following chapters are focusing on editing and manipulation the content of the `ResourceCustomisationData.config` file. All instrument specific functions based on SCPI commands sequences as well as device names and types need to be added here.

### For further reading

- Remote Control and Instrument Drivers
- SCPI-99 Standard

# 3 ResourceCustomisationData.config Structure

The `ResourceCustomisationData.config` file is structured as an XML document. It consists of four main elements as follows:




The different root nodes will be explained in detail in the following chapters and expanded by code examples and command sequences.

## 3.1 Element 'IsEdited'

This field signals the MINX software to parse or ignore the settings of the RCD file at startup depending on the parameter `true` or `false` (default).

After adding own parameters, it is mandatory to also change this field to `true` in order to be recognized by MINX.

**Code example**

<IsEdited>`true`</IsEdited>

## 3.2 Element 'UserFunctions'

Similar to the functions keys on a standard PC, the User Function buttons can be assigned instrument commands. These instrument commands can apply a setting (e.g. turn on RF) or download data from an instrument to the master PC (e.g. screenshot).

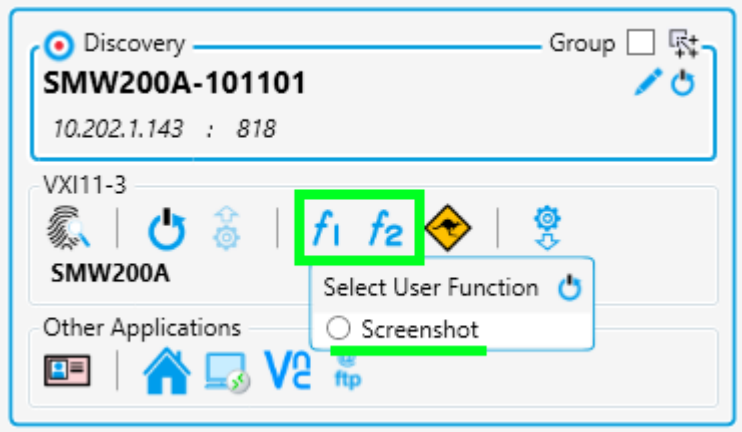



Figure 2: Selecting a user function for a specific instrument

Many of these extended functions already exist in the MINX software. However, adding new functions does not require a MINX software upgrade. A new function can be made available by adding a SCPI commands sequence to an existing or new child of the **UserFunctions** node.

The general structure of a child of **UserFunctions** is as follows:

```
<UserFunction>
  <Models>
    <Model>Device1Name</Model>
    <Model>Device2Name</Model>
    […]
  </Models>
  <Functions>
    <Function>
      <Name>FunctionName</Name>
      <Commands>
        <Command>SCPICommand1</Command>
        <Command>SCPICommand2</Command>
        <Command>SCPICommand3</Command>
        […]
      </Commands>
  </Functions>
</UserFunction>
```

Optional elements are:

- `<IoTimeOut_ms>NumericValue</IoTimeOut_ms>`
- 
```
<ReturnValueSave>
  <FileNames>
    <FileName>Filename.Extension</FileName>
  </FileNames>
</ReturnValueSave>
```

(continued on the following page →)

► 
```
<ReturnValuePreview>
 <PreviewLines>
  <PreviewLine>FreeText1{SiValueData_<SIUnit>}or{ParameterData}</PreviewLine>
  <PreviewLine>FreeText2{SiValueData_<SIUnit>}or{ParameterData}</PreviewLine>
  […]
 </PreviewLines>
 <Monitor>
  <Interval>NumericValue</Interval>
 </Monitor>
</ReturnValuePreview>
```

Each format string between the <PreviewLine /> tags formats a returned instrument value until all returned values are used up.

## Code examples

**UserFunction 'Screenshot' for SWM200A signal generator**

```
<UserFunction>
 <Models>
  <Model>SMW</Model>
  <Model>SMW200A</Model>
 </Models>
 <Functions>
  <Function>
   <Name>Screenshot</Name>
   <IoTimeOut_ms>5000</IoTimeOut_ms>
   <Commands>
    <Command>
       HCOPy:FILE:NAME:AUTO:STATe OFF
    </Command>
    <Command>HCOPy:IMAGe:FORMat PNG</Command>
    <Command>
       HCOPy:FILE:NAME '/var/user/screenshot.png'
    </Command>
    <Command>HCOPy:EXECute</Command>
    <Command>*OPC?</Command>
    <Command>
       MMEMory:DATA? '/var/user/screenshot.png'
    </Command>
   </Commands>
   <ReturnValueSave>
    <FileNames>
     <FileName>screenshot.png</FileName>
    </FileNames>
   </ReturnValueSave>
  </Function>
 </Functions>
</UserFunction>
```

**UserFunction 'Monitor Power' for NRPxxS/N power sensors**

```
<UserFunction>
 <Models>
  <Model>NRP18SN</Model>
  <Model>NRP40S</Model>
 </Models>
 <Functions>
  <Function>
   <Name>Monitor Power</Name>
   <Commands>
    <Command>UNIT:POWer DBM</Command>
    <Command>FORMat:DATA ASCii,3</Command>
    <Command>SENSe:POWer:AVG:FAST ON</Command>
    <Command>INITiate:IMMediate</Command>
    <Command>FETCh:SCALar:POWer?</Command>
   </Commands>
   <ReturnValuePreview>
    <PreviewLines>
     <PreviewLine>
       Power: {SiValueData_dBm}
     </PreviewLine>
    </PreviewLines>
    <Monitor>
     <Interval>1000</Interval>
    </Monitor>
   </ReturnValuePreview>
  </Function>
 </Functions>
</UserFunction>
```

## Recommendation

In order to make sure the SCPI command sequence is working properly and resulting the expected behavior, check it with the MINX SCPI Session terminal in advance. There, SCPI commands can be sent individually or scripted (without flow control). This is the recommended way to debug SCPI sequence, as there is no debugging in the config file and SCPI errors may hang the instrument.

## 3.3 Element 'SaveRecallCommands'

MINX also implements save and recall (SRCL) functions for many instrument models and has dedicated buttons for this. SRCL settings can be applied individually or to an entire saved group. Additionally, such settings can be copied between benches or to all benches having same instruments models.

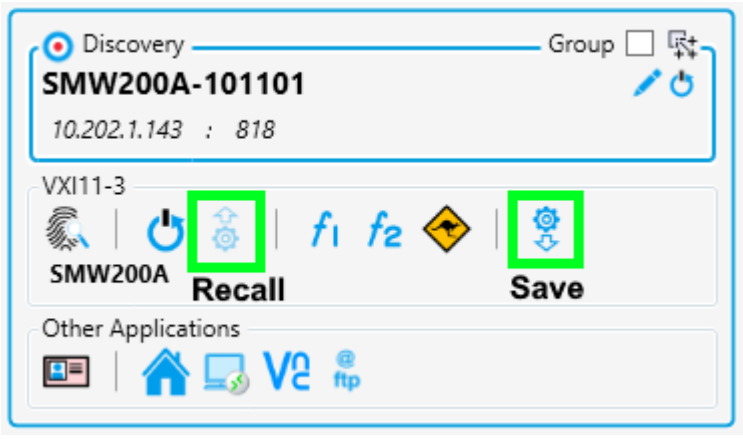



Figure 3: Showing where to find the save and recall functions

A new instrument model specific SRCL function can be implemented by adding as a new child of the **SaveRecallCommands** node.

The general structure of a child of **SaveRecallCommands** is as follows:

```
<SaveRecallCommand>
  <Models>
    <Model>Device1Name</Model>
    <Model>Device2Name</Model>
    […]
  </Models>
  <Commands>
    <Command>
      <Get>
        <Command>SCPICommand1</Command>
        <Command>SCPICommand2</Command>
        <Command>SCPICommand3</Command>
        […]
      </Get>
      <Set>
        <Command>SCPICommand1</Command>
        <Command>SCPICommand2</Command>
        <Command>SCPICommand3</Command>
        […]
      </Set>
    </Command>
  </Commands>
</SaveRecallCommand>
```

Optional elements are:

- <IoTimeOut_ms>NumericValue</IoTimeOut_ms>

## Code examples

**Save/Recall commands for SWM/SMBV signal generators**

```
<SaveRecallCommand>
  <Models>
    <Model>SMW</Model>
    <Model>SMW200A</Model>
    <Model>SMBV100A</Model>
    <Model>SMBV100B</Model>
  </Models>
  <IoTimeOut_ms>10000</IoTimeOut_ms>
  <Commands>
    <Command>
      <Get>
        <Command>*SAV 1</Command>
        <Command>*OPC?</Command>
        <Command>
          MMEMory:STORe:STATe 1,'/var/user/systemsetup.minx'
        </Command>
        <Command>*OPC?</Command>
        <Command>
          MMEMory:DATA? '/var/user/systemsetup.minx'
        </Command>
      </Get>
      <Set>
        <Command>
          MMEMory:DATA '/var/user/systemsetup.minx',#
        </Command>
        <Command>*WAI</Command>
        <Command>
          MMEMory:LOAD:STATe 1,'/var/user/systemsetup.minx'
        </Command>
        <Command>*WAI</Command>
        <Command>*RCL 1</Command>
      </Set>
    </Command>
  </Commands>
</SaveRecallCommand>
```

**Save/Recall command for Value Instruments**

```
<SaveRecallCommand>
  <Models>
    <Model>HMC8012</Model>
    <Model>HMC8015</Model>
    <Model>HMF2550</Model>
    <Model>HMO1022</Model>
    <Model>RTC1002</Model>
    <Model>RTB2002</Model>
    <Model>RTB2004</Model>
  </Models>
  <Commands>
    <Command>
      <Get>
        <Command>SYST:SET?</Command>
      </Get>
      <Set>
        <Command>SYST:SET #</Command>
      </Set>
    </Command>
  </Commands>
</SaveRecallCommand>
```

## Hints

- The ASCII character `'#'` introduces {data block} and indicates that this {block data} will be inserted afterwards. It is a format which is suitable for the transmission of large amounts of data.

  For example, a command using a block data parameter has the following structure:
  `FORMat:READings:DATA #45168xxxxxxxx`

  The next number indicates how many of the following digits describe the length of the data block. In the example the 4 following digits indicate the length to be 5168 bytes. The data bytes follow. During the transmission of these data bytes all end or other control signs are ignored until all bytes are transmitted.

- MINX reads {block data} automatically as part of a 'Save' function call and re-inserts it as part of a 'Recall' operation.

- Use `*OPC?` to wait for <Get /> commands to complete. This allows MINX to wait for the instrument to complete its local dump of saving settings to a file.

- `*WAI` used in the <Set /> paragraph is to enforce the instrument to wait for the settings to be uploaded completely and save to a file on the instrument's disc, before processing the command to apply the settings file.

- `*OPC?` is synchronizing MINX and `*WAI` is synchronizing the instrument. It's important that these two aren't mixed up.

## 3.4 Element 'UploadFunctions'

Similar to UserFunctions, **UploadFunctions** implement instrument commands in order to upload data files to instruments (e.g. firmware updates, arb files).
Upload Functions are also available from the group actions toolbar and are applied to all instruments of the selected model (with its group checkbox checked).

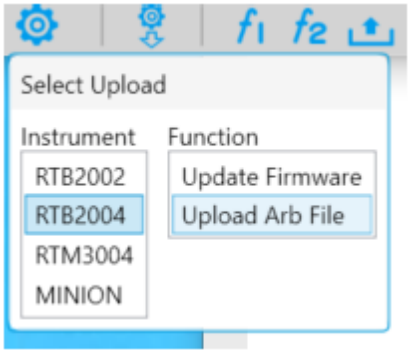



Figure 4: Uploading an arb file for a RTB2004 scope

As in the chapters earlier, data upload functions for any instrument model applicable can be made available by adding a new entry to or editing a child of the **UploadFunctions** node.

The general structure of a child of **UploadFunctions** is as follows:

```
<UploadFunction>
  <Models>
    <Model>Device1Name</Model>
    <Model>Device2Name</Model>
    […]
  </Models>
  <Functions>
    <UFunction>
      <Name>UFunctionName</Name>
      <FileDialogTitle>DialogTitle</FileDialogTitle>
      <FileDialogFilter>FileFilters</FileDialogFilter>
      <Commands>
        <Command>SCPICommand1</Command>
        <Command>SCPICommand2</Command>
        <Command>SCPICommand3</Command>
        […]
      </Commands>
    </UFunction>
  </Functions>
</UploadFunction>
```

### Code examples

**Firmware upload for RTB/RTM scopes**

```
<UploadFunction>
 <Models>
  <Model>RTB2002</Model>
  <Model>RTB2004</Model>
  <Model>RTM3004</Model>
 </Models>
 <Functions>
  <UFunction>
   <Name>Update Firmware</Name>
   <FileDialogTitle>Select Firmware File</FileDialogTitle>
   <FileDialogFilter>
    Firmware Files|*.fwu
   </FileDialogFilter>
   <Commands>
    <Command>
     DIAGnostic:UPDate:TRANsfer:OPEN FIRMware
    </Command>
    <Command>
     DIAGnostic:UPDate:TRANsfer:DATA 0,0,{BlockData}
    </Command>
    <Command>
     DIAGnostic:UPDate:TRANsfer:CLOSe
    </Command>
    <Command>
     DIAGnostic:UPDate:INSTall''
    </Command>
   </Commands>
  </UFunction>
 </Functions>
</UploadFunction>
```

**Arb file upload for RTB/RTM scopes**

```
<UploadFunction>
 <Models>
  <Model>RTB2002</Model>
  <Model>RTB2004</Model>
 </Models>
 <Functions>
  <UFunction>
   <Name>Upload Arb File</Name>
   <FileDialogTitle>Select Arb File</FileDialogTitle>
   <FileDialogFilter>
    CSV Files (*.csv)|*.csv|Binary
    Files (*.trf)|*.trf
   </FileDialogFilter>
   <Commands>
    <Command>
     MMEMory:DATA '/INT/REFERENCE/{FileName}'
     ,{BlockData}
    </Command>
   </Commands>
  </UFunction>
 </Functions>
</UploadFunction>
```

## 3.5 Element 'GenericNames'

This element is mainly used with MINION to display a generic name, e.g. 'Oscilloscope', to a bench instrument. Each string must be a model like returned by a `*IDN?` query and will be used to identify compatible instruments.

The general structure of a child of **GenericNames** is as follows:

```
<GenericName>
 <Name>DeviceCategory</Name>
 <Models>
  <Model>Device1Name</Model>
  <Model>Device2Name</Model>
  <Model>Device3Name</Model>
  […]
 </Models>
</GenericName>
```

## Code examples

| Device category 'Oscilloscope' | Device category 'Spectrum Analyzer' |
|---|---|
| <GenericName><br><Name>Oscilloscope</Name><br><Models><br><Model>RTB2002</Model><br><Model>RTB2004</Model><br><Model>RTM3004</Model><br><Model>RTO</Model><br></Models><br></GenericName> | <GenericName><br><Name>Spectrum Analyzer</Name><br><Models><br><Model>FSW</Model><br><Model>FSV3004</Model><br><Model>FSVR-30</Model><br><Model>FSV-7</Model><br></Models><br></GenericName> |

## Hint

► The <Name /> element is a free-format text string that can be named arbitrarily by the user. However, it's recommended to use commonly agreed device category tags. Please find a recommendation in the following table.

## Recommendation for device category tags

| Device Category Tags | Device examples |
|---|---|
| Arbitrary Waveform Generator | HMF2525, HMF2550 |
| Digital Multimeter | HMC8012 |
| EMI Test Receiver | ESW44. ESR26 |
| Oscilloscope | RTA4004, RTO2004, RTH HMC1002 |
| Power Analyzer | HMC8015 |
| Source Meter Unit | 6240B, 6241A, 6166 |
| Power Sensor | NRP18SN, NRQ6, NRPM3 |
| Power Supply | HMP4040, HMC8043, NGP814, NGL202 |
| Radio Test Set | CMA180, CMS54 |
| Spectrum Analyzer | FSW85, FSVR30, FSVA3044 |
| Signal Generator | SMW200A, SMBV100B, SMA100B, SMC100A |
| System Components | IQW, RSC, OPS320 |
| Vector Network Analyzer | ZNA43, ZVA110, ZNB40 |
| Wireless Communication Tester | CMX500, CMW500, CMW100, CP200 |
| […] | |

# 4 References


[1] RSUK, "MINX - Measurement Instrument Network eXplorer," Rohde & Schwarz UK, [Online]. Available: https://www.rohde-schwarz.com/appnote/gfm344. [Accessed 16 January 2021].

[2] D. Ho, "Notepad++," [Online]. Available: https://notepad-plus-plus.org/downloads/. [Accessed 16 June 2020].

[3] Microsoft Corporation, "XML Notepad 2007," [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=7973. [Accessed 16 June 2020].


# 5 Ordering Information

| Designation | Type | Order No. |
|---|---|---|
| MILS-MINX T&M Laboratory Software | MILS-MINX | 3657.4639.02 |
| MILS-MINION Measurem. Inst. NW Interactive Organisat. Node | MILS-MINION | 3656.7640.02 |

# 6 Appendix

## 6.1 SCPI Session Window

Opens a dialog window providing entry and retrieval of instrument remote control information, using the SCPI instrument command language. Commands can be sent individually or scripted (no flow control).

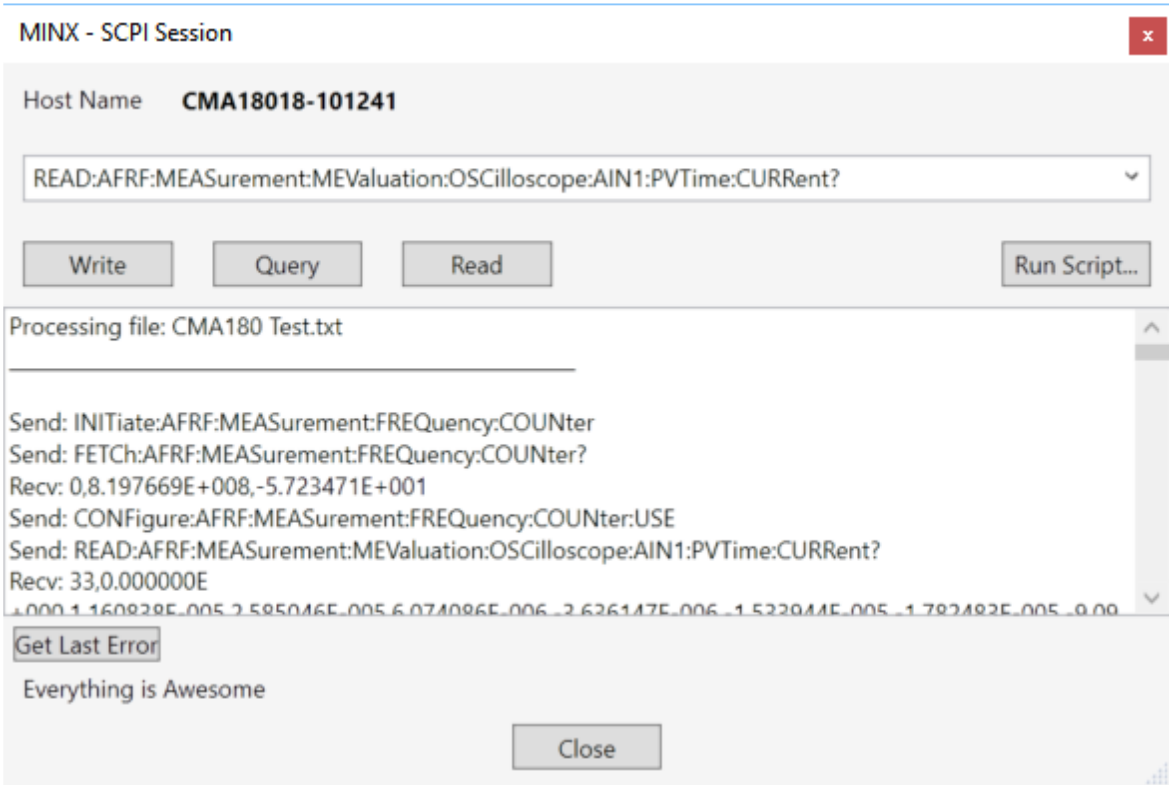



Figure 5: SCPI Session window as part of MINX software for debugging SCPI command sequences

## 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.training.rohde-schwarz.com

PAD-T-M: 3572.7186.00/01.00/EN

## Rohde & Schwarz customer support

www.rohde-schwarz.com/support


GFM344 | Version 0e | 06.2020
Application Note | MINX User-Defined Instrument Functions
Data without tolerance limits is not binding | Subject to change

www.rohde-schwarz.com