MATLAB Toolkit for R&S® Signal Generators

Application Note

Products:

- R&S®SMW200A
- R&S®SMU200A
- R&S®SMJ100A
- R&S®SMATE200A
- R&S®AMU200A
- R&S®SMBV100A
- R&S®AFQ100A
- R&S®AFQ100B
- R&S®SMA100A
- R&S®SMB100A
- R&S®SMC100A

This R&S MATLAB® Toolkit provides routines for remote controlling Rohde & Schwarz Signal Generators from MATLAB® scripts. Additionally, I/Q data can be turned into Rohde & Schwarz waveform files for use with the instrument's ARB option. This application note describes the installation and use of this R&S MATLAB® Toolkit on Microsoft Widows and Linux based systems.

T. Röder, C. Tröster

24.4.2013

Application Note

ROHDE&SCHWARZ

Table of Contents

1. Notices

The following abbreviations are used throughout this application note.

- The R&S® MATLAB® Toolkit for R&S® Signal Generators is referred to as MATLAB Toolkit or toolkit.
- The R&S® SMW200A Vector Signal Generator is referred to as SMW.
- The R&S® SMU200A Vector Signal Generator is referred to as SMU.
- The R&S® SMJ100A Vector Signal Generator is referred to as SMJ.
- The R&S® SMATE200A Vector Signal Generator is referred to as SMATE.
- The R&S® SMBV100A Vector Signal Generator is referred to as SMBV.
- The R&S® AFQ100A I/Q Modulation Generator is referred to as AFQ.
- The R&S® AFQ100B UWB Signal and I/Q Modulation Generator is referred to as AFQ.
- The R&S® AMU200A Baseband Signal Generator and Fading Simulator is referred to as AMU.

2. Overview

MATLAB is widely used for the simulation of communication systems and the creation or analysis of custom signals. This toolkit provides functions that simplify the remote control of Rohde & Schwarz instruments from MATLAB scripts. These functions cover the most common tasks, such as sending SCPI commands to the instrument and reading responses back from it. Additional functions convert I/Q data into R&S waveform files and transfer these files to the instrument's ARB. A set of example scripts demonstrate the use of the toolkit functions in various applications. The toolkit supports remote control via GPIB hardware, raw TCP/IP socket connections, or the VISA (Virtual Instrument Software Architecture) interface.

3. Features

The toolkit provides the following functionality.

- Support Microsoft Windows and Linux based systems
- Open device connections to R&S Signal Generators
- Send SCPI commands or queries
- Process a script with SCPI commands or queries
- Create an R&S waveform file from IQ data and send the waveform file to the instrument
- Access instruments from within Agilent ADS (Advanced Design System)

4. System Requirements

1.2 Hardware

- The MATLAB requirements (CPU, memory, hard drive)
- For the instrument remote control (depending on instrument)
  - 100MBit LAN
  - GPIB hardware
  - USB interface

1.3 Software

The MATLAB Toolkit can be used on Microsoft Windows XP or Linux operating systems. The following configuration was tested by Rohde & Schwarz.

- Microsoft Windows XP based system
  - Windows XP, Service Pack 2
  - MATLAB 7.4.0 R2007a
  - MATLAB Instrument Control Toolbox
  - National Instruments VISA Version 4.0

- Linux based system
  - Linux kernel 2.6.18, e.g. Open SuSE 10.2
  - MATLAB R2009a
    (MATLAB Instrument Control Toolbox provided from R2009a on)
  - National Instruments VISA Version 4.1

5. Windows XP Installation

1.4 Unpacking The Files

The MATLAB Toolkit comes as a set of files bundled in a ZIP archive.

'RS_MATLAB_Toolkit_<version number>.zip

First, create a new folder under your MATLAB toolbox directory, e.g.

```
C:\Program Files\MATLAB\R2007a\toolbox\RsMatlabToolkit
```

The contents of the archive should now be unpacked into this directory.

1.5 Setting The Path In MATLAB

For a convenient use of the toolkit functions it is required to add the installation path of the toolkit scripts to the MATLAB environment. This can be done by selecting *File → Set Path* from the menu bar. This brings up the 'Set Path' dialog where the new path is added using the Add Folder… button.

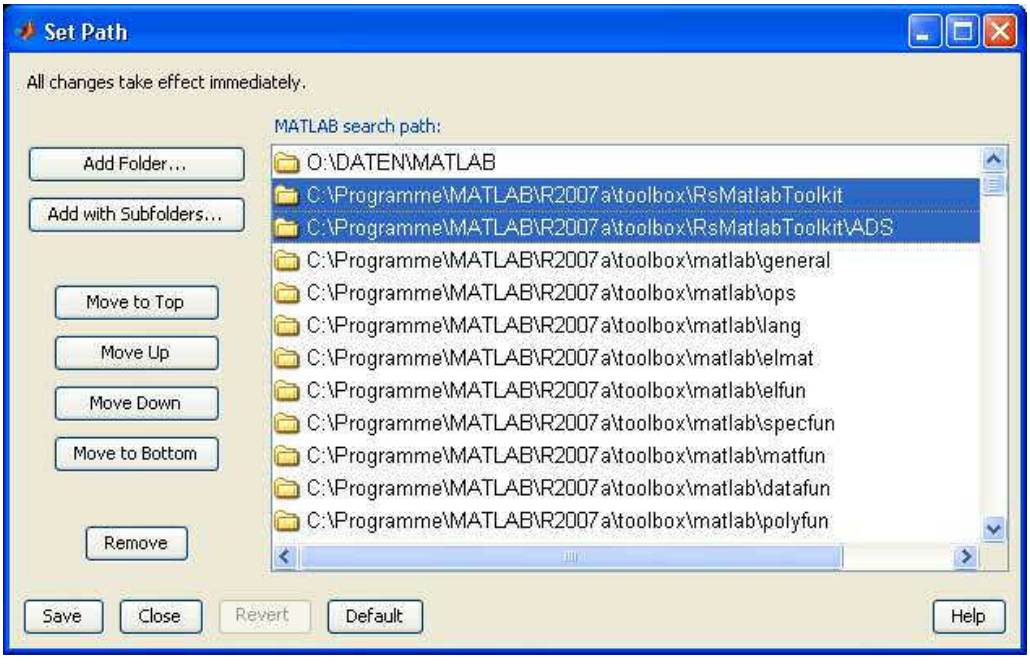


Figure 1 MATLAB Set Path dialog

Note: When using MATLAB 7.0 (R14) it is required to remove unused instrument drivers. This is a confirmed problem in MATLAB 7.0 (R14) in the way that MATLAB interfaces with GPIB cards.

To remove the drivers please open the directory

```
\toolbox\instrument\instrumentadaptors\win32
```

in the MATLAB directory, create a new directory called backup and move every dll except mwnigpib.dll and mwnivisa.dll to the backup directory.

6. Linux Installation

Starting with version R2009a MATLAB provides the Instrument Control Toolbox on Linux installation. Using these toolbox scripts is recommended for all instrument communication because it allows writing platform independent code and simplifies instrument access greatly.

1.6 Prerequisites

- Linux distribution with kernel sources and symbols installed
- VISA installation package, e.g. from National Instruments

1.7 Unpacking The Files

The toolkit comes as a set of files bundled in a ZIP archive. Extract all files into a path in your MATLAB toolbox directory.

```
# md /opt/matlab2009/toolbox/RsMatlabToolkit

# unzip <archive.zip> \
      -d /opt/matlab2006/toolbox/RsMatlabToolkit
```

After the files are unpacked follow the Step ‘Set the path to the toolkit in MATLAB’ from the Windows installation procedure.

1.8 VISA Installation

This brief guide was developed for an Open SuSE 10.2 installation and the National Instruments VISA package. Please see the VISA web site (www.ni.com/visa) for download and details about license and installation.

First, prepare your kernel sources as root user.

```
# cd /usr/src/linux
# make cloneconfig
# make prepare
```

Next, mount the National Instruments ISO image into a folder.

```
# mkdir /media/visa
# mount -t iso9660 -o loop,ro <imagefile> /media/visa
# cd /media/visa
```

Run the National Instruments installer and make sure to add the ‘development’ option.

```
# ./INSTALL
```

You should now have a working VISA installation. Verify your installation by invoking the NIvisaic or visaconf tools.

```
# NIvisaic
# visaconf
```

1.9 USB Support

In order to use USB for VISA remote control the current kernel must support usbfs or on older kernels usbdevfs. Verify that this file system is mounted by using the mount command.

```
# mount | grep usb
```

If usbfs is mounted a line, such as ‘usbfs on /proc/bus/usb type usbfs …’ should appear.
By default only the root user is allowed for raw access to USB devices. In order to allow access to normal users National Instruments provides a script that configures your system accordingly. Execute this script as root user.

```
# /usr/local/vxipnp/linux/NIvisa/USB/
      AddUsbRawPermissions.sh
```

Enter the vendor ID 0x0AAD for Rohde & Schwarz and the product of your instrument. A list of all product IDs can be found in the next chapter.

Now connect the instrument to the USB port.

7. VISA Resource Strings

The general format of the USB VISA resource string is

```
USB::<vendor id>::<product id>::<serial number>::INSTR
```

The vendor ID for all Rohde & Schwarz devices is 0x0AAD. The product ID depends on the instrument and can be taken from the list below.

USB Device IDs	
Instrument	**Device ID**
SMW200A	0x92
AFQ100A	0x4B
AFQ100B	0x66
AMU200A	0x55
SMATE200A	0x46
SMBV100A	0x5F
SMF100A	0x47
SMA100A	0x48
SMB100A	0x54
SMC100A	0x6E

The list represents all R&S Signal Generators that support USB remote control at the time of writing.

8. R&S Toolkit Functions

The toolkit provides various scripts for basic instrument access. The following list summarizes these functions. Each function is located in a MATLAB script using the name of the function.

Function Overview	
Function / .m File	**Description**
rs_connect.m	Create GPIB or VISA object for an instrument or open TCP/IP socket connection and verify the link
rs_check_instrument_errors.m	Query the instrument's error queue by repeatedly sending SYST:ERR?
rs_send_command.m	Send a SCPI command to the instrument
rs_send_query.m	Send a SCPI command to the instrument and query the result
rs_batch_interpret.m	Process a file containing SCPI commands
rs_generate_wave.m	Generate waveform file from complex vector and transfer to instrument
rs_visualize.m	Visualizes I/Q data by generating a time domain, FFT, and I/Q plane plot.

The next paragraph explains the syntax and use of these functions.

1.10 rs_connect.m

The function rs_connect() sets up the instrument connection and tests the link. It returns a handle to the remote controlled instrument. The connection can be established using the GPIB or VISA interface as well as via a raw TCP/IP socket connection. The recommended interface type is VISA.

```
rs_connect ('gpib',     <'advantech|agilent|cec|contec|ics|
                        iotech|keithley|mcc|ni'>,
                        <board number>, <primary address>
                        [, <secondary address>])

rs_connect ('visa',     '<ni|agilent|tek>',
                        '<visa resource string>')

rs_connect ('tcpip',    '<hostname>')
```

GPIB parameters

<board number>	GPIB board number (usually 0)
<primary address>	GPIB bus address of the instrument
<secondary address>	The secondary GPIB bus address of the instrument (optional)

VISA parameters

<VISA resource string>	The VISA resource string describes the device as well as the interface type.

TCP/IP parameters

<hostname>	The instrument host name or the IP address

Return values

<status >	1 if successful
<object >	The handle to the instrument

The GPIB and VISA functions require an identifier for the installed hard- or software interface. The function call for opening device number 28 connected to the first National Instruments GPIB card is

```
>> [err, instrObj] = rs_connect( 'gpib', 'ni', 0, 28 );
```

VISA offers more flexibility over direct GPIB connections as it allows the use of different interface types. Please consult your Rohde & Schwarz instrument manual or data sheet for details about which remote control interface is supported by your instrument.

Please see the documentation of your VISA installation for details about the VISA resource strings.

Note:
LAN based remote control requires that the instrument's firewall is disabled. This, for example, applies to the Microsoft Windows based instruments SMU, SMJ, and SMATE. The Linux based instruments, such as the SMBV do not have an internal firewall.

1.11 rs_check_instrument_errors.m

The function rs_check_instrument_errors() sends SYST:ERR? queries to the instrument until the error queue is entirely cleared.

```
<status> = rs_check_instrument_errors( <object> );
```

Parameters

<object>	The instrument object returned by rs_connect()

Return value

<status>	1 if successful

Example code

```
>> [err, instrObj] = rs_connect( 'gpib', 'ni', 0, 28 );
>> err = rs_send_command( instrObj, 'XXX' );
>> err = rs_check_instrument_errors( instrObj );
-113,"Undefined header;XXX"
```

1.12 rs_send_comand.m

The function rs_send_command() sends a single SCPI command to a previously connected instrument.

```
<status> = rs_send_command( <object>, '<command>' );
```

Parameters

<object>	The instrument object returned by rs_connect()
<command>	SCPI command, e.g. 'FREQ 1.2GHz'

Return value

<status>	1 if successful

Example code

```
>> [err, instrObj] = rs_connect( 'gpib', 'ni', 0, 28 );
>> err = rs_send_command( instrObj, '*RST' );
```

1.13 rs_send_query.m

The function rs_send_query() works similar to rs_send_command() with the exception that an answer from the instrument is expected after the command has been sent.

```
[<status>, <answer>] = rs_send_query( <object>, '<command>' );
```

Parameters

<object>	The instrument object returned by rs_connect()
<command>	SCPI command, e.g. `'*IDN?'`

Return values

<status>	1 if successful
<answer>	Contains the answer from the instrument

Example code

```
>> [err, instrObj] = rs_connect( 'gpib', 'ni', 0, 28 );
>> [err, answer] = rs_send_query( instrObj, '*IDN?' );
```

1.14 rs_batch_interpret.m

This function processes a series of SCPI commands or queries from a text file.

```
[<status>, <answer>] = rs_batch_interpret( <object>,
                                           '<batch-file>' );
```

Parameters

<object>	The instrument object returned by rs_connect()
<batch-file>	Path and name of the batch-file

Return values

<status>	1 if successful
<answer>	Contains the queries results in a structure answer(x).text, where x is a consecutive number of the queries, starting with the index one. The highest number is the total number of queries.

Example batch file (scpi.txt)

```
% Comment line
*IDN?
FREQ 1.2 GHz
POW -10.0 dBm
OUTP:STAT ON
SYST:ERR?
```

Example code

```
>> [err, answer] = rs_batch_interpret( instrObj, \
                                        'scpi.txt' );
```

answer(1).text contains the return information from '*IDN?'
answer(2).text contains the return information from 'SYST:ERR?'

1.15 rs_generate_wave.m

This function builds a waveform file from an I/Q vector. It also sends the generated file to the instrument's mass memory system and optionally starts the ARB in path A or B.

```
[<status>] = rs_generate_wave( <object>, <struct>,
                               <playback>, <save_local> );
```

Parameters

<object >	The instrument object returned by rs_connect() If this number is set to 0 the waveform is only stored locally and not sent to the instrument.
<struct>	I/Q data and waveform parameters
<playback>	0 = ARB is not started after transfer 1 = waveform is activated in path A 2 = waveform is activated in path B
<save_local>	0 = waveform is not stored on local PC 1 = waveform is stored on local PC (current directory)

Return value

<status>	1 if successful

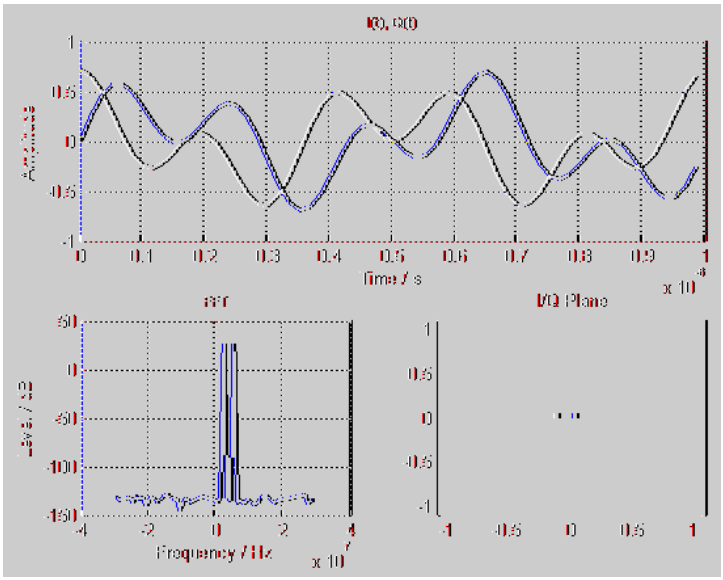
The <struct> format

I_data	1D array of I values	(mandatory)
Q_data	1D array of Q values	(mandatory)
markerlist.one	2D array marker list e.g. Position Value 0 0 10 1 50 0 -> [[0 0];[10 1];[50 0]]	(optional)
markerlist.two	2D array marker list	(optional)
markerlist.three	2D array marker list	(optional)
markerlist.four	2D array marker list	(optional)
clock	desired clock rate in Hz	(mandatory)
path	storage path in the remote device including drive letter: e.g. "D:\Files"	(mandatory)
filename	waveform name in the remote device file extension ".wv" is mandatory	(mandatory)
comment	Comment	(optional)

Markers are digital output signals that are generated synchronous to the signal output. These signals can be used for synchronization of other devices. For more information please see the instrument's user manual.

1.16 rs_visualize.m

This function plots the I/Q values.




```
[<status>] = rs_visualize( FSample, I_data, Q_data_);
```

Parameters

<FSample>	The sample rate in Hz
<I_data>	1D array with I values
<Q_data>	1D array with Q values

Return value

<status>	1 if successful

9. Example Scripts

The example scripts demonstrate the use of the toolkit functions. The following list provides a brief overview of the examples.

Script Overview	
Script / .m File	**Functionality**
Connect.m	Connect to instrument and send *IDN? and *OPT? This example also describes the various hardware interfaces in detail.
Create_IQ_AWGN.m	Create a noise (AWGN) signal and load to the instrument's ARB.
Create_IQ_Chirp.m	Create a frequency sweep signal and load to the instrument's ARB.
Create_IQ_MultiCarrier.m	Create a multi carrier signal and load to the instrument's ARB.
Create_IQ_Pulse.m	Create a pulsed signal and load to the instrument's ARB.
Create_IQ_TwoTone.m	Create a two tone signal and load to the instrument's ARB.
Convert_Mat2Wv.m	Read data from a .mat file and convert into a Rohde & Schwarz waveform file for use with the instruments ARB
Run_SCPI_Batch.m	Process a list of SCPI commands that are read from a text file

10. ADS Support

The ADS sub directory of the toolkit contains a MATLAB routine that can be called from ADS in order to convert a complex vector into a Rohde & Schwarz ARB waveform file. The MATLAB routine also allows the user to start the waveform playback in path A or B and set the RF frequency and level.

1.17 Installation

On Microsoft Windows systems ADS only requires that the PATH environment variable is set to the \bin\win32 sub directory of the MATLAB installation.

```
PATH=<matlabroot>\bin\win32;%PATH%
```

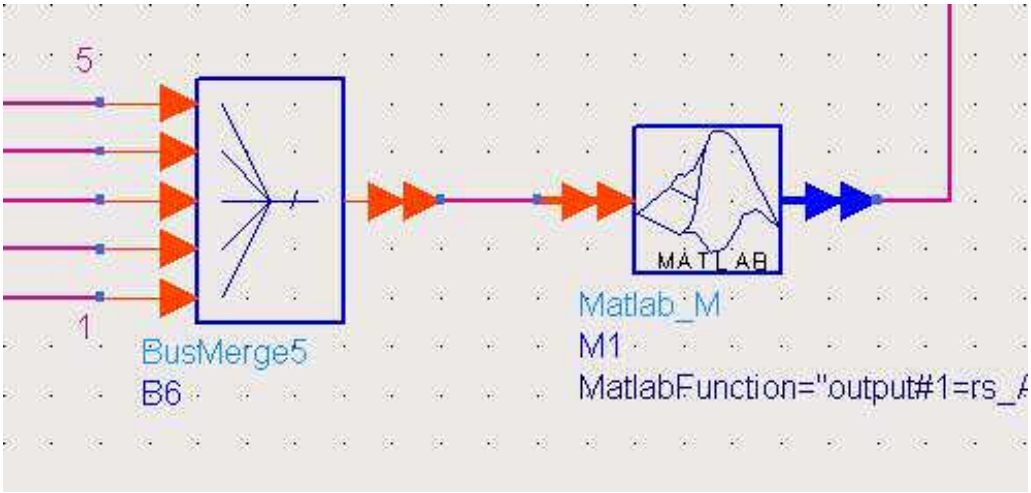
Additionally, it must be ensured that all toolkit scripts can be found from MATLAB.

1.18 Usage

The first step is to place a Matlab_M object into your schematic. This object directly runs a MATLAB .m script during a simulation.

Next, a BusMerge5 object is required to combine the input parameters for the R&S Matlab_M object.

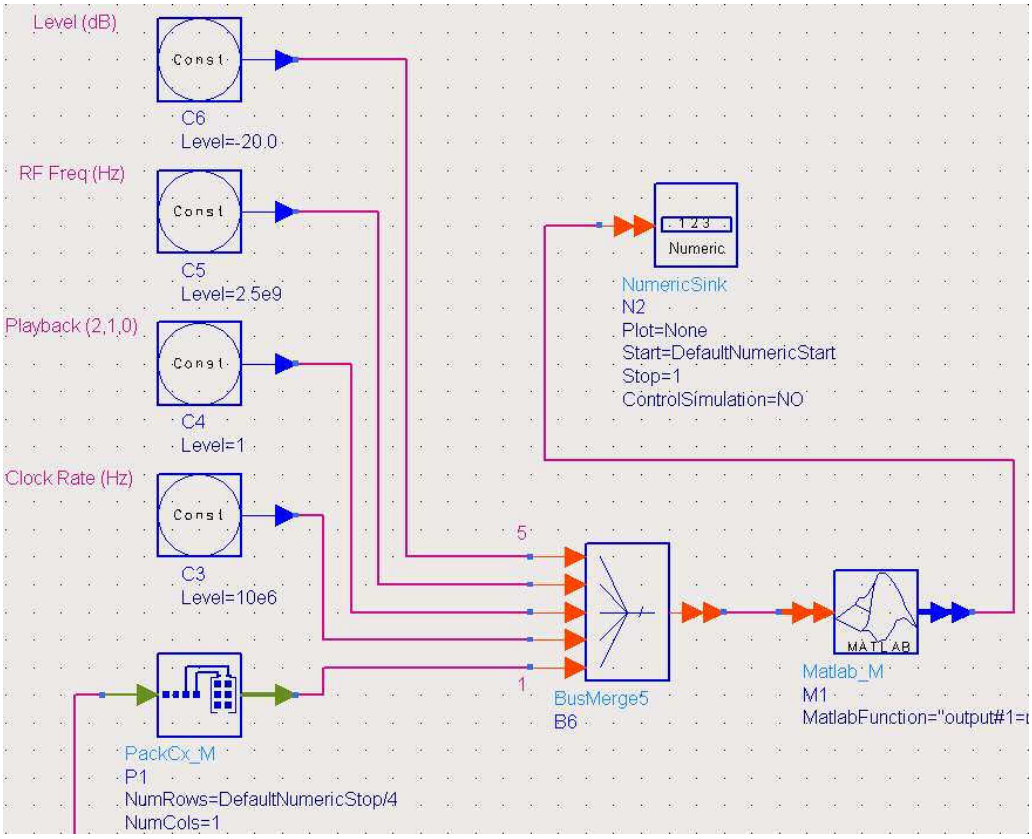
Connect the two objects as shown below and edit the Matlab_M properties according to the list below.




```
Script Directory   =
MatlabSetUp        =
MatlabFunction     = output#1=rs_ADS_IQSink( input#1, input#2,
                   input#3, input#4, input#5,
                   'GPIB0::28::INSTR', 'D:/', 'test.wv',
                   'comment', 'copyright')
MatlabWrapUp       =
```

The property MatlabFunction defines the call to MATLAB as well as all input and return parameters. The placeholder input# is used for one of the parameters passed to the BusMerge5 object. In the example above the VISA resource string is set to a device connected to the primary GPIB board and set to address 28. The MATLAB routine creates a waveform file locally and copies it to the specified location under the given name (D:\test.wv in the example above).

Finally, the input values need to be provided to the BusMerge5 block as indicated in the following picture. Const objects may be used for this purpose.



It is also required to pack the input data stream into a vector. This can be done by using the *PackCx_M* object from ADS.
The example above uses QPSK encoded data and thus the length of the vector is set to DefaultNumericStop/4 (no oversampling used, DefaultNumericStop sets number of input bits).

The list below outlines the input parameters to the BusMerge block.

- Complex vector of I/Q data
- Clock rate in Hz
- Playback path for instrument 0=none, 1=A, 2=B
- RF Frequency in Hz
- RF Level in Hz

The MATLAB script rs_ADS_IQSink() performs the following actions when called with the above parameters.

- Open the VISA based instrument connection
- Process complex vector and turn into local waveform file
- Transfer the waveform file block wise to the instrument
- Start playback on instrument
- Set RF parameters

- Close the instrument connection

The function uses the National Instruments VISA interface by default. The MATLAB code needs to be changed if other interfaces are used.

11. Additional Information

1GP61	R&S MATLAB® Toolkit for R&S® NRP-Z Senors
1MA171	How to use Rohde & Schwarz Instruments in MATLAB®
1EF46	Using MATLAB® for Remote Control and Data Capture with R&S Spectrum and Network Analyzers
1EF51	Using R&S Signal, Spectrum and Network Analyzers with MATLAB®

About Rohde & Schwarz
Rohde & Schwarz is an independent group of companies specializing in electronics. It is a leading supplier of solutions in the fields of test and measurement, broadcasting, radiomonitoring and radiolocation, as well as secure communications. Established more than 75 years ago, Rohde & Schwarz has a global presence and a dedicated service network in over 70 countries. Company headquarters are in Munich, Germany.

Environmental commitment
- Energy-efficient products
- Continuous improvement in environmental sustainability
- ISO 14001-certified environmental management system

Certified Quality System
ISO 9001

Regional contact
Europe, Africa, Middle East
+49 89 4129 137 74
customersupport@rohde-schwarz.com

North America
1-888-TEST-RSA (1-888-837-8772)
customer.support@rsa.rohde-schwarz.com

Latin America
+1-410-910-7988
customersupport.la@rohde-schwarz.com

Asia/Pacific
+65 65 13 04 88
customersupport.asia@rohde-schwarz.com

Rohde & Schwarz GmbH & Co. KG
Mühldorfstraße 15 | D - 81671 München
Phone + 49 89 4129 - 0 | Fax + 49 89 4129 – 13777

www.rohde-schwarz.com