



Products: R&S® SMU200A, R&S® SMJ100A, R&S® SMATE200A, R&S® AFQ100A, R&S® AMU200A

# R&S® Matlab Transfer Toolbox for R&S® Vector Signal Generators and R&S® Modulation Generators

## Application Note

The Matlab Transfer Toolbox allows the conversion of complex I/Q vectors into the Rohde & Schwarz waveform file format. The resulting waveform files can be sent to the instrument ARB using a GPIB or TCP/IP connection. This application note describes the installation and use of the toolbox on Microsoft Windows and Linux based systems.



## Contents

1	Notices .....	3
2	Overview .....	3
3	Features .....	3
4	Hardware and Software Requirements .....	4
	Instrument Requirements .....	4
	PC Hardware Requirements .....	4
	PC Software Requirements .....	5
5	Release Notes .....	5
6	Windows XP Installation .....	5
	Unpacking the files .....	5
	Set the path to the transfer toolbox in MATLAB .....	6
7	Linux Installation .....	6
	Unpacking the files .....	6
	Prerequisites .....	7
	Installation of VISA .....	7
	Compiling the gateway driver .....	7
	USB Support (AFQ100A) .....	8
8	USB Device VISA Resource Strings .....	8
9	Using the Toolbox on Windows Based Systems .....	9
	Function overview .....	9
	Demo scripts .....	9
	Function Description .....	10
	rs_connect () .....	10
	rs_send_command () .....	11
	rs_send_query () .....	11
	rs_batch_interpret () .....	11
	rs_generate_wave () .....	12
10	Using the Toolbox on Linux Based Systems .....	13
	Function overview .....	13
	Demo scripts .....	13
	Function Description .....	14
	rs_LinuxVisaDriver () .....	14
	rs_Lin_generate_wave () .....	15
	rs_Lin_batch_interpret () .....	15
11	ADS Support .....	16
	Installation .....	16
	Usage .....	16
12	References .....	18
13	Ordering information .....	18

## 1 Notices

The following abbreviations are used throughout the application note for R&S test equipment:

- The vector signal generator R&S® SMU200A is referred to as SMU
- The vector signal generator R&S® SMJ100A is referred to as SMJ
- The vector signal generator R&S® SMATE200A is referred to as SMATE
- The I/Q Modulation Generator R&S® AFQ100A is referred to as AFQ
- The Baseband Signal Generator And Fading Simulator R&S® AMU200A is referred to as AMU

Microsoft®, Windows®, MS Windows®, Windows NT®, and MS-DOS® are U.S. registered trademarks of Microsoft Corporation.

MATLAB is a U.S. registered trademark of The Math Works, Inc.

Agilent® and Agilent® EEs of ADS are registered trademarks of Agilent Technologies

Rohde & Schwarz® is a registered trademark of Rohde & Schwarz GmbH & Co. KG.

## 2 Overview

MATLAB is widely used for the simulation of communication systems as well as for the creation or analysis of custom signals. The R&S Matlab Transfer Toolbox provides a link between the MATLAB environment and the Rohde & Schwarz signal generators, allowing the user to turn complex vectors into ARB waveforms.

This toolbox provides functions for the most common tasks, such as communicating with the instrument or the data transfer. The link to the instrument is either established via GPIB or by using the VISA (Virtual Instrument Software Architecture) interface.

## 3 Features

The R&S Matlab Transfer Toolbox provides the following functionality:

- Use on Microsoft Windows and Linux based systems
- Interface to Agilent ADS (Advanced Design System)
- Open a device connection by either using GPIB or VISA
- Send SCPI commands or queries
- Process a script of SCPI commands or queries
- Create an R&S waveform file from complex vectors and send the waveform file to the instrument
- Select the baseband path and optionally activate waveform playback
- Demo files are included

## 4 Hardware and Software Requirements

### Instrument Requirements

Instrument	Options
R&S <sup>®</sup> SMU200A	R&S <sup>®</sup> SMU-B9 (128 Msamples) or R&S <sup>®</sup> SMU-B10 (64 Msamples) or R&S <sup>®</sup> SMU-B11 (16 Msamples) R&S <sup>®</sup> SMU-B13
R&S <sup>®</sup> SMJ100A	R&S <sup>®</sup> SMJ-B9 (128 Msamples) or R&S <sup>®</sup> SMJ-B10 (64 Msamples) or R&S <sup>®</sup> SMJ-B11 (16 Msamples) R&S <sup>®</sup> SMJ-B13
R&S <sup>®</sup> SMATE200A	R&S <sup>®</sup> SMATE-B9 (128 Msamples) or R&S <sup>®</sup> SMATE-B10 (64 Msamples) or R&S <sup>®</sup> SMATE-B11 (16 Msamples) R&S <sup>®</sup> SMATE-B13
R&S <sup>®</sup> AMU200A	R&S <sup>®</sup> SMATE-B9 (128 Msamples) or R&S <sup>®</sup> SMATE-B10 (64 Msamples) or R&S <sup>®</sup> SMATE-B11 (16 Msamples) R&S <sup>®</sup> SMATE-B13
R&S <sup>®</sup> AFQ100A	R&S <sup>®</sup> AFQ-B10 (256 Msamples) R&S <sup>®</sup> AFQ-B11 (1 Gsamples)

### PC Hardware Requirements

Besides the MATLAB requirements it is suggested to have a physical remote control link to the instrument established. This could either be a LAN interface or GPIB hardware. The AFQ also supports remote control by the USB interface.

## PC Software Requirements

The R&S Matlab Transfer Toolbox can be used on Microsoft Windows XP or Linux operating systems. The following configuration is tested and recommended by Rohde & Schwarz:

### 1. Windows XP

- Windows XP, Service Pack 2
- MATLAB 7.4.0 R2007a
- National Instruments VISA Version 4.0

### 2. Linux

- Linux kernel 2.6.18, e.g. Open SuSE 10.2
- MATLAB R2006a
- National Instruments VISA Version 4.1

## 5 Release Notes

- |       |   |
|-------|---|
| V 1.0 | - first release   |
| V 1.1 | - corrections to rs_generate_wave<br>- added rs_import_binary<br>- additional demo routines |
| V 2.0 | - entirely redesigned function interface<br>- Linux support added<br>- ADS support added    |

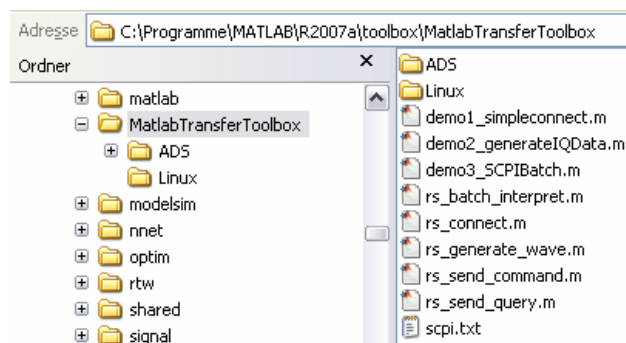
## 6 Windows XP Installation

### Unpacking the files

The Matlab Transfer Toolbox comes as a set of files bundled in a ZIP archive.

*'Matlab Transfer Toolbox\_<version number>.zip*

It is recommended to extract all files into a path, such as  
C:\Program Files\MATLAB\R2007a\toolbox\MatlabTransferToolbox.

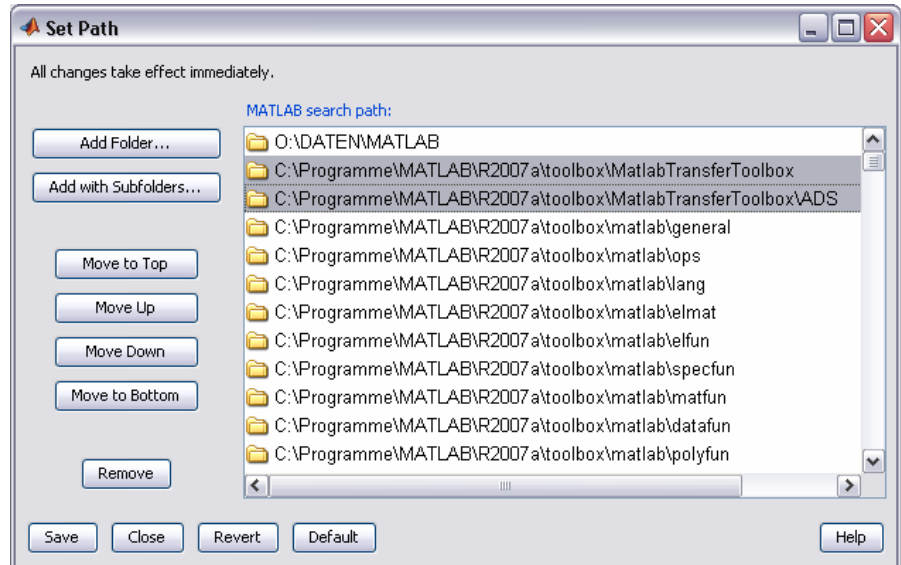


(path on German Windows XP installation)

### Set the path to the transfer toolbox in MATLAB

For a convenient use of the toolbox functions it is required to add the installation path of the Matlab Transfer Toolbox 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 with the *Add Folder...* button.



**Note:** *If you use MATLAB 7.0 (R14) you need to remove unused instrument drivers. This is a confirmed bug 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.*

*More information can be found under:*

[www.mathworks.com/support/solutions/data/1-PJ6VF.html](http://www.mathworks.com/support/solutions/data/1-PJ6VF.html)

## 7 Linux Installation

At the time of writing there is no direct VISA support available from within MATLAB. To overcome this issue the R&S Matlab Transfer Toolbox contains a C gateway routine that builds the link between Matlab and VISA or GPIB. The C gateway routine is available as source code and needs to be compiled on the target system using the MATLAB 'mex' command.

### Unpacking the files

The Matlab Transfer Toolbox comes as a set of files bundled in a ZIP archive. The ZIP archive contains the Windows and Linux files. For Linux it is only required to use the files from the Linux sub directory.

Extract all linux related files into a path in your MATLAB toolbox directory.

## Linux Installation

---

```
# md /opt/matlab2006/toolbox/MatlabTransferToolbox
# unzip <archive.zip> "Linux/*.*"
    -d /opt/matlab2006/toolbox/MatlabTransferToolbox
```

After the files are unpacked follow the Step 'Set the path to the transfer toolbox in MATLAB' from the Windows installation procedure.

### Prerequisites

You need to have a Linux distribution with kernel sources and symbols installed. Additionally, a C-compiler environment such as gcc needs to be available to compile the gateway.

### Installation of VISA

The first step should be the installation of the VISA drivers. This brief guide is valid for Open SuSE 10.2 and the National Instruments VISA. Please see their web site ([www.ni.com/visa](http://www.ni.com/visa)) for download and more details about license and installation.

Prepare your kernel as root user:

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

Now mount the National Instruments ISO image to start the installation:

```
# mkdir /tmp/iso
# mount -t iso9660 -o loop,ro
    <imagefile> /tmp/iso
# cd /mnt/iso
```

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 Nlvisaic or visaconf tools.

```
# Nlvisaic
# visaconf
```

### Compiling the gateway driver

The gateway driver must be compiled from within MATLAB. Before attempting to compile the driver it is required to locate the visa library on your PC. This can be done by using the Linux find command:

```
# find / -name libvisa.so
```

By default the path is /usr/local/vxipnp/linux/lib. Note this path and start MATLAB. Now change to the installation directory of the Matlab Transfer Toolbox. Execute the mex command below to compile the gateway:

```
mex -L /usr/local/vxipnp/linux/lib
    -l visa rs_LinuxVisaDriver.c
```

If no error messages appear you are ready to use the example files from the Matlab Transfer Toolbox.

### USB Support (AFQ100A)


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 to configure your system accordingly. Execute this script as root:

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



Use the vendor ID 0x0aad for Rohde & Schwarz and the product ID 0x004b for the AFQ100A instrument.

Now connect the instrument to the USB port.

## 8 USB Device VISA Resource Strings

The general format of the USB VISA resource string is:

```
USB::::<product id>::::INSTR
```

For all Rohde & Schwarz devices the vendor ID is 0x0aad. The product ID depends on the instrument type and can be taken from the list below.

AFQ-100A	0x004b
NRP-Z21	0x0003
NRP-FU	0x0004
FSH-Z1	0x000b
NRP-Z11	0x000c
NRP-Z22	0x0013
NRP-Z23	0x0014
NRP-Z24	0x0015
NRP-Z51	0x0016
NRP-Z52	0x0017
NRP-Z55	0x0018
FSH-Z18	0x001a
NRP-Z91	0x0021
NRP-Z81	0x0023
NRP-Z37	0x002d
NRP-Z27	0x002f



## 9 Using the Toolbox on Windows Based Systems

### Function overview

Function / .m file	Description
rs_connect ()	Create GPIB or VISA object for your device and verify the connection
rs_send_command ()	Send a SCPI command to your device
rs_send_query ()	Send a SCPI query to your device
rs_batch_interpret ()	Process a list of SCPI command
rs_generate_wave ()	Generate waveform file from complex vector and transfer to instrument

### Demo scripts

MATLAB script file	Functionality
demo1_simpleconnect.m (connect to instrument)	<ul style="list-style-type: none"> <li>- Connect to instrument</li> <li>- Send *IDN?</li> <li>- Receive result</li> <li>- Close connection</li> </ul>
demo2_generateIQData.m (generates FM chirp signal)	<ul style="list-style-type: none"> <li>- Connect to instrument</li> <li>- Send *IDN? and verify instrument</li> <li>- Create FM chirp and build waveform</li> <li>- Transfer waveform to instrument</li> <li>- Close connection</li> </ul>
demo3_SCPIBatch.m (process SCPI commands)	<ul style="list-style-type: none"> <li>- Connect to instrument</li> <li>- Process a file with SCPI commands</li> <li>- Close connection</li> </ul>
Demo4_NRPSweep.m (sweep across frequency)	<ul style="list-style-type: none"> <li>- Connect to instrument</li> <li>- Run frequency seep and capture level with NRPZ sensor connected to sensor input of generator</li> <li>- Close connection</li> </ul>

### Function Description

#### rs\_connect ()

The function `rs_connect()` is used to setup the device connection and test the link. It returns a handle to the remote instrument. The device can be connected either via GPIB or the VISA interface.

Command syntax:

```
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>')
```

#### GPIB:

board number	GPIB board number (usually 0)
primary address	GPIB bus address of your device
secondary address	The secondary GPIB bus address of your device (optional)

#### VISA:

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

#### Return values

status	1 if successful
object	Handle to your instrument

Both function types require an identifier for the installed hard- or software interface. A typical function call for opening device number 28 connected to a National Instruments GPIB card is therefore `'rs_connect ('gpib', 'ni', 0, 28)'`. VISA offers more flexibility over direct GPIB connections as it allows different interface types. Currently Rohde & Schwarz instruments provide GPIB and LAN remote control capabilities. The AFQ100A may also be remote controlled using the USB interface.

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

**Note:** *If you use a TCP/IP link, please ensure that the firewall on your instrument is switched off otherwise remote controlling the instrument is not possible.*

### **rs\_send\_command ()**

The function `rs_send_command()` is used to send a single SCPI command to a previously connected instrument.

#### *Usage*

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

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

#### *Return value*

status	1 if successful
--------	-----------------

### **rs\_send\_query ()**

The function `rs_send_query()` works similar to `rs_send_command()` with the exception that an answer from the instrument is read back.

#### *Usage*

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

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

#### *Return value*

status	1 if successful
answer	Contains the answer from the instrument

### **rs\_batch\_interpret ()**

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

#### *Usage*

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

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

#### *Return value*

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

Example of a batch file:

```
% Comment line
*IDN?
FREQ 50 MHz
POW 7.3 dBm
OUTP:STAT ON
SYST:ERR?
```

After calling the command with

```
[status, answer] = rs_batch_interpret (object, 'SCPIfile.txt');
```

answer(1).text contains the return information from '\*IDN?'

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

### **rs\_generate\_wave ()**

This function build a R&S waveform file from MATLAB I/Q data. In addition, it sends the generated file to the instrument and starts the ARB playback in path A or B.

#### *Usage*

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

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 played in path A 2 = waveform is played 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 format of struct is as follows:*

I_data	1D array of I values	(mandatory)
Q_data	1D array of Q values	(mandatory)
markerlist.one	2D array marker list	(optional)
	e.g.   Position       Value	
	0            0	
	10          1	
	50          0	
	⇒[[0 0];[10 1];[50 0]]	
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)

Marker lists can be used for synchronization of other instruments or as indication for certain parts of a waveform. For more information please see the user manual or online help of your instrument.

## 10 Using the Toolbox on Linux Based Systems

Using the R&S Matlab Transfer Toolbox on Linux based systems is slightly different from the Windows version. The reason is that a gateway is used instead of the functions provided by the MathWorks Instrument Control Toolbox. At the time of writing the MATLAB functions do not support VISA or GPIB communication without 3<sup>rd</sup> party driver software.

The gateways driver provided with the R&S Matlab Transfer Toolbox does only support the VISA communication and functionality is limited to basic read- and write-commands.

### Function overview

Function / .m file	Description
rs_LinuxVisaDriver ()	Gateway routine for VISA functions
rs_Lin_generate_wave ()	Generate waveform file from complex vector and transfer to instrument
rs_Lin_batch_interpret ()	Process a list of SCPI command

### Demo scripts

MATLAB script file	Functionality
demo1_simpleconnect.m	<ul style="list-style-type: none"><li>- Connect to instrument</li><li>- Send *IDN?</li><li>- Receive result</li><li>- Close connection</li></ul>
demo2_generateIQData.m (generates FM chirp signal)	<ul style="list-style-type: none"><li>- Connect to instrument</li><li>- Send *IDN? and verify instrument</li><li>- Create FM chirp and build waveform</li><li>- Transfer waveform to instrument</li><li>- Close connection</li></ul>
demo3_SCPIBatch.m	<ul style="list-style-type: none"><li>- Connect to instrument</li><li>- Process a file with SCPI commands</li><li>- Close connection</li></ul>

### Function Description

#### rs\_LinuxVisaDriver ()

This function is the gateway function and builds the link between MATLAB and VISA.

General command syntax:

```
rs_LinuxVisaDriver ('command', arg...)
```

#### Open the connection

```
index = rs_LinuxVisaDriver ('open', '<VISA rsc>');
```

index                      session index, value < 0 in case of error

VISA rsc                    VISA resource string

The open function opens and tests an instrument connection. Currently only the VISA interface is supported and thus the second function parameter must be a valid VISA resource string.

The return value is an index that needs to be used for all further function calls related to this instrument. When the driver is unloaded all open connections are automatically closed.

#### Close the connection

```
status = rs_LinuxVisaDriver ('close', index);
```

status                      return status (TRUE or FALSE)

index                        session index as returned from *open*

The close function closes the link to a previously opened instrument connection. After calling the close function the index number becomes invalid and cannot be used any further.

#### Send SCPI command to instrument

```
status = rs_LinuxVisaDriver ('write', index,  
                                          '<command>');
```

status                      return status (TRUE or FALSE)

index                        session index as returned from *open*

command                     SCPI command string

This function sends a command an instrument that is identified by the index number obtained from a previous *open* call. Commands are SCPI instructions that do not return any value from the instrument. However, one exception exists for commands terminated on '\*OPC?'. If this string is prevalent at the end of a command the function automatically executes a read and evaluates the result.

### Send query to instrument

```
[status, result] = rs_LinuxVisaDriver ('read',  
                                     index, '<command>');
```

status	return status (TRUE or FALSE)
result	return string from instrument
index	session index as returned from <i>open</i>
command	SCPI command string

This function sends a command to the instrument and then waits for the device response. The response string is returned.

The default timeout is set in the gateway routine and cannot be changed by a function call (5 seconds).

### rs\_Lin\_generate\_wave ()

```
[status] = rs_Lin_generate_wave (index, struct,  
                                playback, save_local);
```

index	session index as returned from <i>open</i>
struct	I/Q data and waveform parameters
playback	0 = ARB is not started after transfer 1 = waveform is played in path A 2 = waveform is played in path B
save_local	0 = waveform is not stored on local PC 1 = waveform is stored on local PC (current directory)

This function assembles a Rohde & Schwarz waveform file from a complex vector. Additionally, the file can be transferred to the instrument and playback started in one of the ARBs.

### rs\_Lin\_batch\_interpret ()

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

index	session index as returned from <i>open</i>
batch-file	path and name of the batch-file
status	1 if successful
answer	Contains the queries results in a structure <code>answer(x).text</code> , where <code>x</code> is a consecutive number of the queries, starting with the index one. The highest number is the total number of queries.

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

## 11 ADS Support

The ADS sub directory of the toolbox 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.

### 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%
```

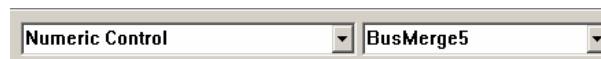
Additionally, it must be ensured that all toolbox scripts can be found from MATLAB.

### Usage

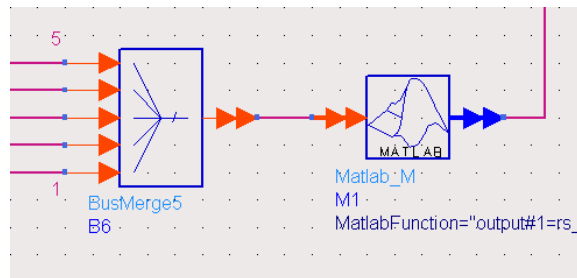
The first step is to place a *Matlab\_M* object into your schematic. This object directly runs a MATLAB .m script during a simulation.



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')
```

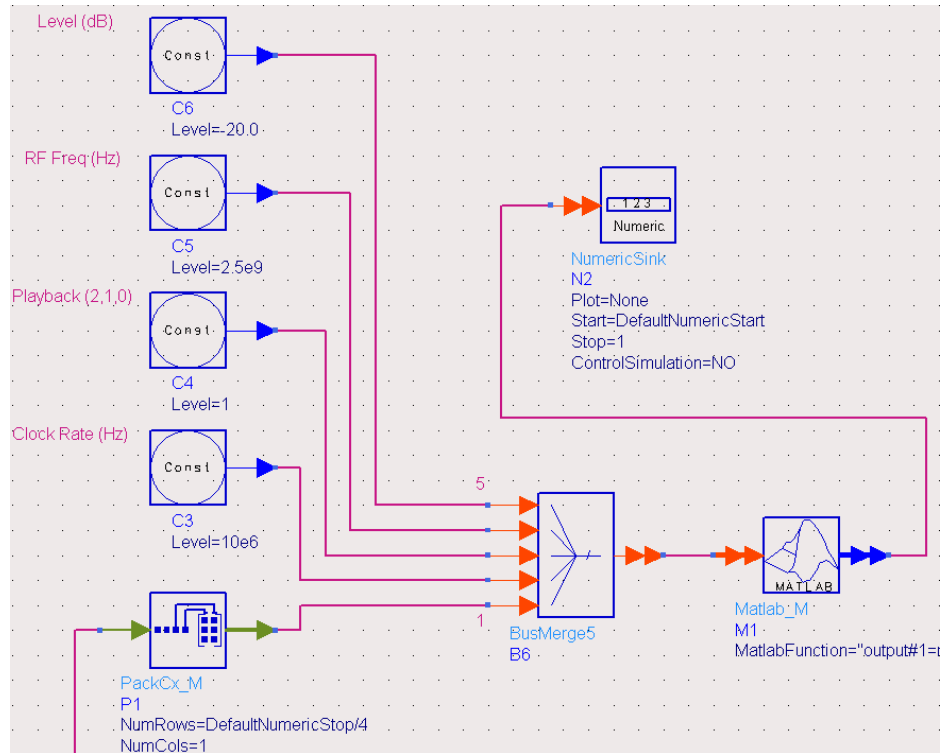
MatlabWrapUp =

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  $\text{DefaultNumericStop}/4$  (no oversampling used,  $\text{DefaultNumericStop}$  sets number of input bits).

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

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

## References

---

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.

## 12 References

- [1] R&S® SMU200A website: [www.rohde-schwarz.com/product/SMU](http://www.rohde-schwarz.com/product/SMU)
- [2] R&S® SMJ100A website: [www.rohde-schwarz.com/product/SMJ](http://www.rohde-schwarz.com/product/SMJ)
- [3] R&S® SMATE200A website: [www.rohde-schwarz.com/product/SMATE](http://www.rohde-schwarz.com/product/SMATE)
- [4] R&S® AFQ100A website: [www.rohde-schwarz.com/product/AFQ](http://www.rohde-schwarz.com/product/AFQ)
- [5] R&S® AFQ100A website: [www.rohde-schwarz.com/product/AMU](http://www.rohde-schwarz.com/product/AMU)
- [6] Transfer Toolbox website: [www.rohde-schwarz.com/appnote/1GP60](http://www.rohde-schwarz.com/appnote/1GP60)

## 13 Ordering information

Type of instrument		
<b>R&amp;S® SMU200A</b>	Vector Signal Generator	1141.2005K02
R&S® SMU-B102	Frequency option 2.2 GHz, 1 <sup>st</sup> RF path	1141.8503.02
R&S® SMU-B103	Frequency option 3 GHz, 1 <sup>st</sup> RF path	1141.8603.02
R&S® SMU-B104	Frequency option 4 GHz, 1 <sup>st</sup> RF path	1141.8603.02
R&S® SMU-B106	Frequency option 6 GHz, 1 <sup>st</sup> RF path	1141.8803.02
R&S® SMU-B202	Frequency option 2.2 GHz, 2 <sup>nd</sup> RF path	1141.9400.02
R&S® SMU-B203	Frequency option 3 GHz, 2 <sup>nd</sup> RF path	1141.9500.02
R&S® SMU-B13	Baseband Main Module	1141.8003.04
R&S® SMU-B10	Baseband Generator with ARB (64 Msamples)	1141.7007.02
R&S® SMU-B11	Baseband Generator with ARB (16 Msamples)	1159.8411.02
<b>R&amp;S® SMJ100A</b>	Vector Signal Generator	1403.4507K02
R&S® SMJ-B103	Frequency option 3 GHz	1403.8502.02
R&S® SMJ-B106	Frequency option 6 GHz	1403.8702.02
R&S® SMJ-B13	Baseband Main Module	1403.9109.02
R&S® SMJ-B10	Baseband Generator with ARB (64 Msamples)	1403.8902.02
R&S® SMJ-B11	Baseband Generator with ARB (16 Msamples)	1403.9009.02
<b>R&amp;S® SMATE200A</b>	Vector Signal Generator	1400.7005K02
R&S® SMATE-B103	Frequency option 3 GHz, 1 <sup>st</sup> RF path	1401.1000.02
R&S® SMATE-B106	Frequency option 6 GHz, 1 <sup>st</sup> RF path	1401.1200.02
R&S® SMATE-B203	Frequency option 3 GHz, 2 <sup>nd</sup> RF path	1401.1400.02
R&S® SMATE-B206	Frequency option 6 GHz, 2 <sup>nd</sup> RF path	1401.1600.02
R&S® SMATE-B13	Baseband Main Module	1401.2907.02
R&S® SMATE-B10	Baseband Generator with ARB (64 Msamples)	1401.2707.02
R&S® SMATE-B11	Baseband Generator with ARB (16 Msamples)	1401.2807.02



ROHDE & SCHWARZ GmbH & Co. KG · Mühl Dorfstraße 15 · D-81671 München · Postfach 80 14 69 · D-81614 München ·  
Tel (089) 4129 -0 · Fax (089) 4129 - 13777 · Internet: <http://www.rohde-schwarz.com>

*This application note and the supplied programs may only be used subject to the conditions of use set forth in the download area of the Rohde & Schwarz website.*