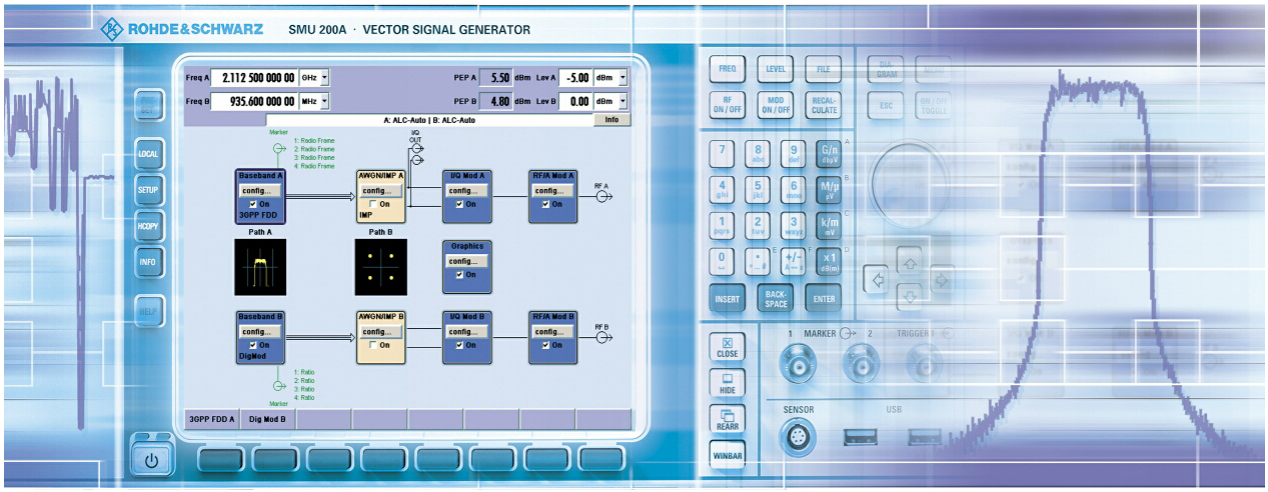


Products: R&S® SMU200A, R&S® SMATE200A, R&S® SMJ100A, R&S® AFQ100A, R&S® AMU200A

Importing Data in ARB, Custom Digital Modulation and RF List Mode

Application Note

The foregoing application note is intended for everyone who desires to become more familiar with the use and upload of custom data in ARB, digital modulation and RF list mode. Particularly, this note describes the instruments R&S® SMU200A, R&S® SMJ100A, R&S® SMATE200A, R&S® AFQ100A and R&S® AMU200A and provides an overview of already available solutions as well as it gives hints for the writing of own applications.

ROHDE&SCHWARZ

Subject to change – Thomas Roeder Mar 2007 – 1GP62_2E

Table Of Contents

1 Overview 3

2 Hardware and Software Requirements 4

3 Features and Operating Modes 5

- Memory Sizes and Playback Rates 6
- The IQ ARB Mode 7
- The Custom Digital Modulation Mode 9
- Custom Mappings 11
- RF List Mode 12
- Using Markers 13

4 Description of File Formats 14

- Waveform Files 15
- Data List Files 17
- Control List Files 18
- RF List Files 19

5 Converting and Uploading Existing Data 20

- ASCII Lists 21
- Matlab Data 22
- Generating Waveforms with R&S WinIQSIM™ 24
- FM/AM/PM with R&S FM/AM ARB software 26
- Using Captured Data from a Spectrum Analyzer 27
- Useful GPIB Commands 28
- Communicating using VISA 32

6 Transferring Data to the Instrument 34

- GPIB Remote Control 34
- USB Remote Control 35
- TCP/IP (VXI-11) Remote Control 35
- Using Shared Drives 36
- Manual Transfer with USB Sticks 37

7 The ARB Toolbox 38

- Example: How to generate an FM chirp signal 39
- Installing the ARB Toolbox 43
- Using the ARBToolbox 44
- Running SCPI batch scripts 45
- Working with the ARB Panel 46
- Working with Custom Digital Modulation 48
- Working with the RF List Mode Panel 50

8 Code Examples 52

- Mat Lab Example for Use with Matlab Toolbox 52
- Lab Windows/CVI C Code examples 54

9 Literature and References 61

10 Additional Information 62

11 Ordering information 62

1 Overview

The foregoing application note is intended for everyone who desires to become more familiar with the use and upload of custom data in ARB, digital modulation and RF list mode. Particularly, this note describes the instruments R&S® SMU200A, R&S® SMJ100A and R&S® SMATE200A and provides an overview of already available solutions as well as it gives hints for the writing of own applications.

Chapter 3 provides a high level overview of the available instrument modes. It briefly describes the theory of operation, the capabilities and limitations that may need to be considered.

Chapter 4 describes all file formats that are used with the ARB and custom digital modulation mode. This chapter is particularly important for people who desire to create their own wave form or data files and upload them to the instrument.

Chapter 5 describes different off the shelf solutions if custom data already exists and shall be converted and uploaded to the instrument.

Chapter 6 describes how data is transferred to the instrument in general.

Chapter 7 introduces the ARB Toolbox, a program that comes with this application note. The program is intended for hands on and provides certain data import options.

Chapter 8 finally provides some code examples that could be used when writing own applications.

Note

The following abbreviations are used throughout this application note:

- SMU for Vector Signal Generator R&S® SMU200A
- SMJ for Vector Signal Generator R&S® SMJ100A
- SMATE for Vector Signal Generator R&S® SMATE200A
- AFQ for I/Q Modulation Generator R&S® AFQ100A
- AMU Baseband Signal Generator and Fading Simulator R&S® AMU200A
- R&S® for Rohde & Schwarz GmbH & Co KG

Trademarks

2 Hardware and Software Requirements

The hardware and software requirements listed below are with respect of the use of the ARB Toolbox that comes with this application note.

Instrument Requirements

Instrument	Option
R&S® SMU200A	R&S® SMU-B11, B10 or B9
R&S® SMJ100A	R&S® SMJ-B11, B10 or B9
R&S® SMATE200A	R&S® SMATE-B11, B10 or B9
R&S® AFQ100A	R&S® AFQ-B9 or B10
R&S® AMU200A	R&S® AMU-B11, B10 or B9
USB Memory Stick	

PC Hardware Requirements

	Minimum	Recommended
CPU	CPU 1 GHz	CPU 2 GHz or higher
RAM	128 MByte	256 MByte
Harddisc	10 MByte free space	50 MByte free space
Monitor	SVGA 800x600	XGA 1024x768 or better
LAN	10 MBit LAN	100 MBit LAN
IEEE Bus	VISA compatible	VISA compatible

PC Software Requirements

	Minimum	Recommended
OS	Windows NT 4.0,2000,XP	Windows XP
Other SW	---	Matlab MCR V7 WinIQSIM™ 4.1 IQ Wizard 4.47
IEEE Bus	Driver V 2.20	>= Driver V 2.20
VISA	Version 3.1	>= Version 3.1
NI CVI	Run Time Version 6.0	>= Run Time Version 6.0

3 Features and Operating Modes

The Rohde & Schwarz vector signal generators SMU, SMJ and SMATE provide a wide range of solutions for the generation and playback of custom data or waveforms.

Most common to many signal generators is the arbitrary mode where IQ samples are played back from the internal memory. The IQ values are usually calculated upfront and loaded onto the unit. This mode is referred to as the ARB mode in this application note.

If digital data shall be used instead of raw IQ samples the instruments offer a data list processing mode. In this mode all data is fed into the internal modulator and can be used with any available modulation scheme such as BPSK, QPSK, QAM, etc. In addition, more sophisticated functions can be applied such as triggers, markers, burst ramping and areas of attenuated RF level. These signals can be used for triggering external instruments as well as for synchronizing multiple base band channels. Custom modulation schemes can be applied to extend the large selection of mappings.

Finally, there is an RF list mode which allows to vary the RF frequency as well as the output level. This happens independently of any base band processing and thus can be used with CW as well as faded or modulated signals. Another benefit is the simple format of frequency lists and the larger frequency deviations than in the ARB mode.

The following paragraph describes all the available modes in more detail.

Memory Sizes and Playback Rates

Rohde & Schwarz currently offers a choice of up to three memory configurations for its base band generators:

SMU200A, SMJ100A, SMATE200A

SMx-B9	512 MB, Base band generator with 128 Msamples, 16 bit
SMx-B10	256 MB, Base band generator with 64 Msamples, 16 bit
SMx-B11	64 MB, Base band generator with 16 Msamples, 16 bit

AFQ100A

AFQ-B11	4 GB, Base band generator with 1 Gsamples, 16 bit
AFQ-B10	1 GB, Base band generator with 256 Msamples, 16 bit

AMU200A

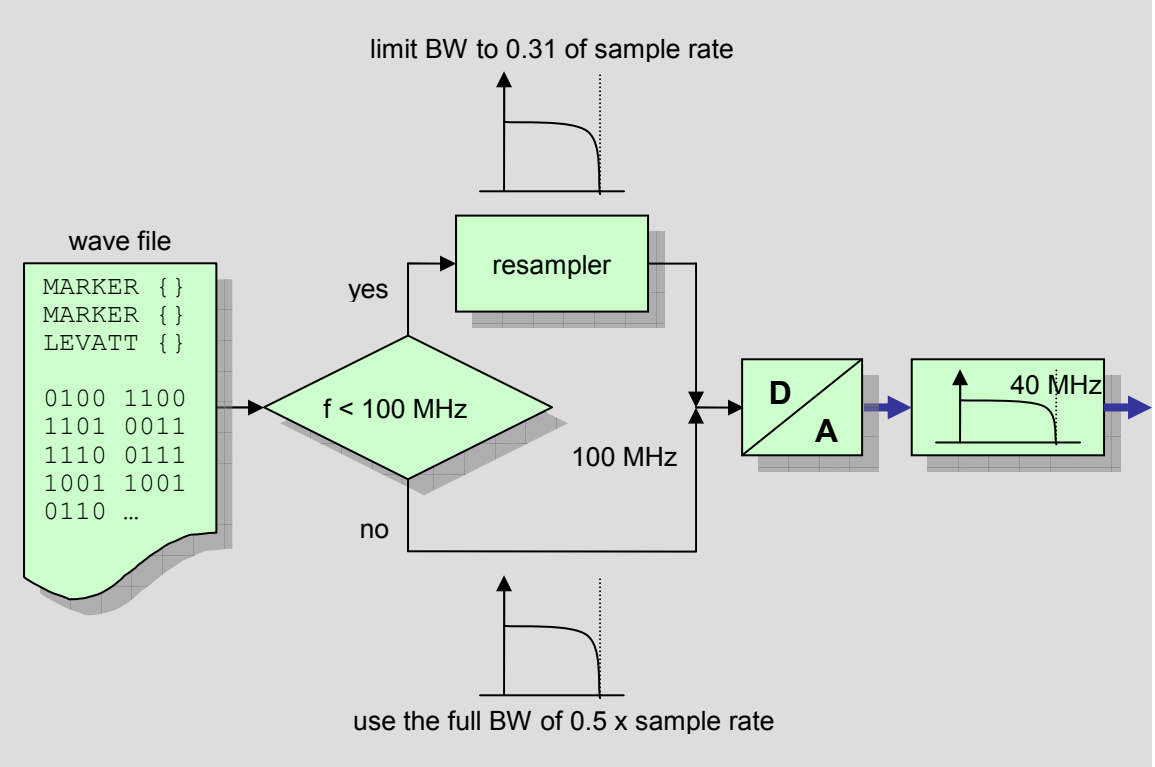
AMU-B9	512 MB, Base band generator with 128 Msamples, 16 bit
AMU-B10	256 MB, Base band generator with 64 Msamples, 16 bit
AMU-B11	64 MB, Base band generator with 16 Msamples, 16 bit

The IQ arbitrary mode uses raw samples which are stored as 16 bit values (16 bit I and 16 bit Q). The memory depth of the options is the true number of complex samples, thus representing the available number of sample points.

If software defined markers are used an additional amount of 4 bits are required per IQ data pair. Thus, a complex sample now consists of 4.5 bytes and reduces the maximum waveform memory to 56 Msamples for B10 and 14 Msamples for B11.

The IQ ARB Mode

The ARB is implemented as an interpolating ARB generator. This means that any played back signal is always re-sampled to 100 MHz and then output.



Block diagram of the ARB generator

The picture illustrates the schematic block diagram of the ARB generator.

All IQ data that is played back at sample rates below 100 MHz is passed on to the resampler. This stage interpolates to an output rate of 100 MHz and limits the useful bandwidth to 0.31 of the sample rate. The signal is then routed into the DAC and subsequently low pass filtered as well as bandwidth limited to 40 MHz.

Arbitrary IQ data can be played back using a clock rate between 400 Hz and 100 MHz. The resolution on the clock rate is 0.001 Hz.

The layout of the ARB system must be taken into consideration when designing waveforms, setting oversampling ratios and clock rates. An example demonstrates the relationships:

The WCDMA standard requires a root raised cosine (RRC) filter with a roll off factor of α = 0.22 to limit the modulation bandwidth to 0.61 times the symbol rate.

$$Bandwidth = \frac{1+\alpha}{2} \cdot Symbolrate$$

Passing the IQ signal on to the resampler stage sets the sample rate requirement to:

$$Samplerate \cdot 0.31 \geq Bandwidth \qquad Samplerate = Symbolrate \cdot Oversampling$$

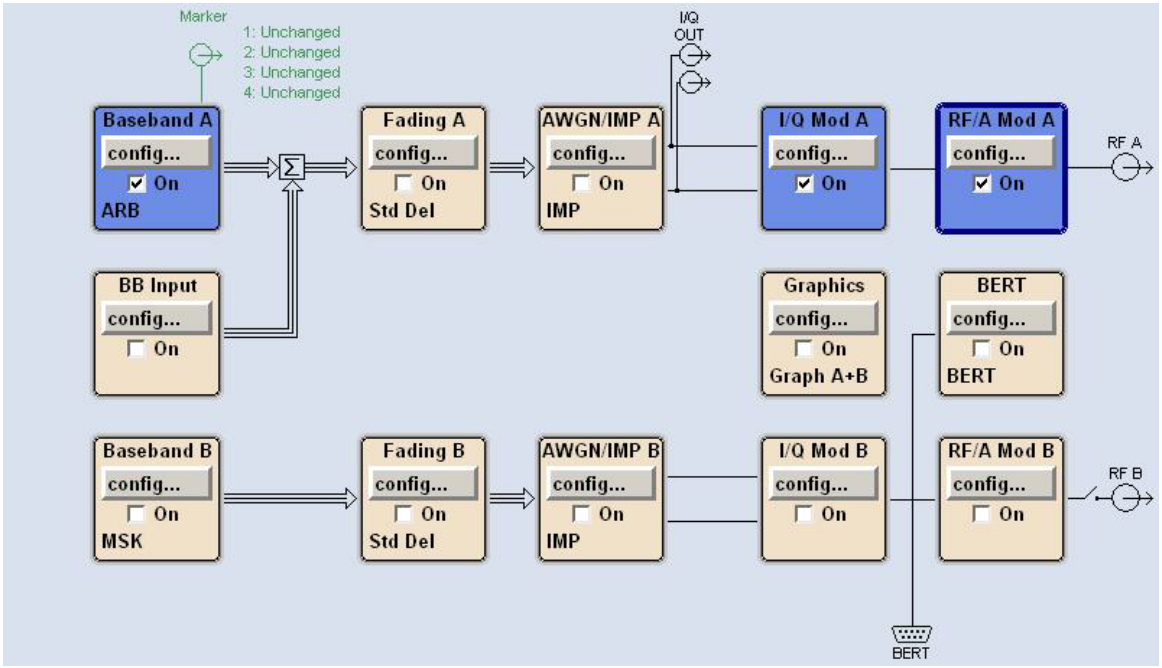
The resulting oversampling ratio is therefore defined as:

$$Oversampling \geq \frac{Bandwidth}{Symbolrate \cdot 0.31}$$

In the above example the modulation bandwidth requirement is 0.61 times the symbol rate which leads to an oversampling ratio of 1.97.

If 10 frames should be played back with 38400 chips each a minimum of 10 * 38400 * 1.97 = 756480 samples are needed. The chip rate required for UMTS is 3.84 Mcps.

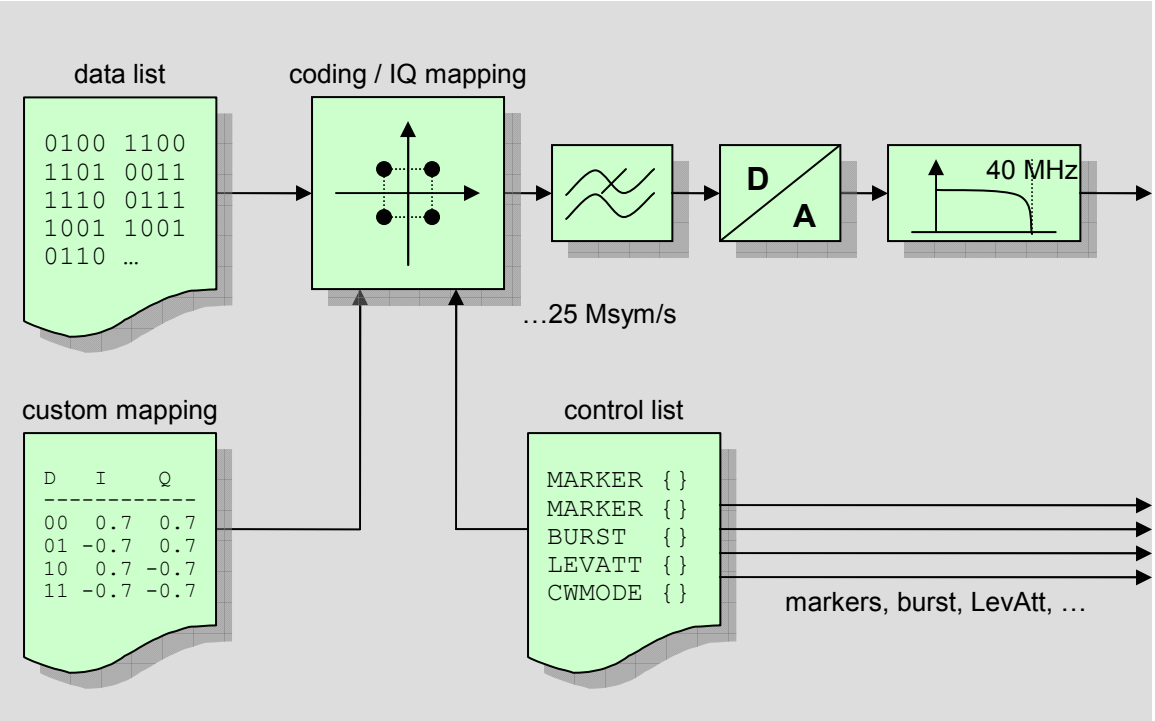
An additional benefit of the concept is that the ARB signal is routed through the fading, noise and impairment stages before it is fed into the modulator and converted into RF. The following picture shows the signal flow in an ARB example.



Screen shot from signal generator

The Custom Digital Modulation Mode

The SMU, SMJ and SMATE can generate digital modulation signals from binary data with user defined characteristics. In contrast to the ARB mode data is processed using either an internal mapping or a custom mapping and then output. The following picture shows an overview of the various lists and their relationships.



Block diagram of custom digital modulation

The data list provides the actual data that is to be played back. The IQ mapping selection (BPSK, QPSK, QAM, etc.) determines the number of bits that are required to form one symbol. Thus, a data list containing useful data (not noise or all ones, zeros) may need to be used with its associated mapping configuration. Customized IQ configurations can be uploaded to the instrument to extend the large list of internally provided mappings. These mappings use *.vam files and can be created by a Matlab application called Mapping Wizard.

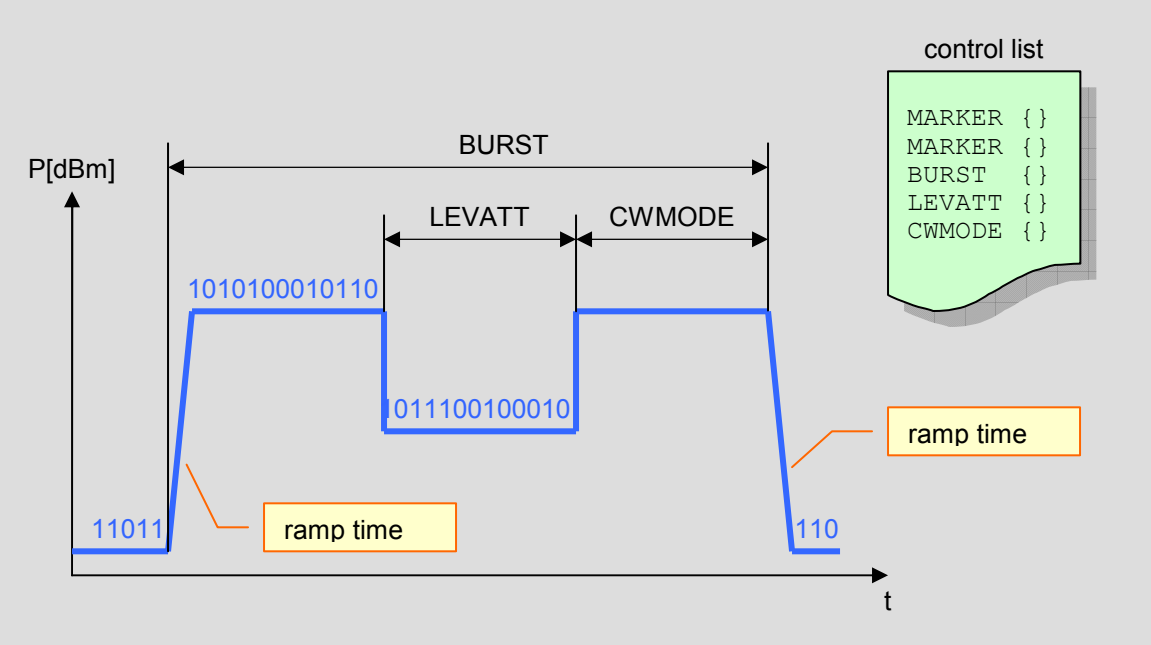
The rate of which data can be output strongly depends on the chosen IQ mapping. The maximum symbol rate ranges between 15 and 25 Msymbols/s.

Mode	Files
ASK / PSK	25 M symbols/s
QAM	25 M symbols/s
MSK	20 M symbols/s
FSK	15 M symbols/s

In addition to the standard IQ mappings coding schemes according to predefined standards such as GSM, EDGE, NADC, WCDMA, etc. can be activated.

Control lists allow to further modify the playback of the encoded and modulated data. Areas of lower signal level, CW sections as well as bursts can be defined. Up to four additional marker signals can be activated or deactivated at any data position and may serve as trigger for external instruments or for synchronization of multiple basebands.

In case burst structures are used there are additional instrument parameters that control the ramping shape, such as rise and fall times as well as delays. The following picture shows how burst areas are defined by the use of a control list. However, the control list only marks the begin and end of a burst or attenuated area. It does not control the rise and fall times.



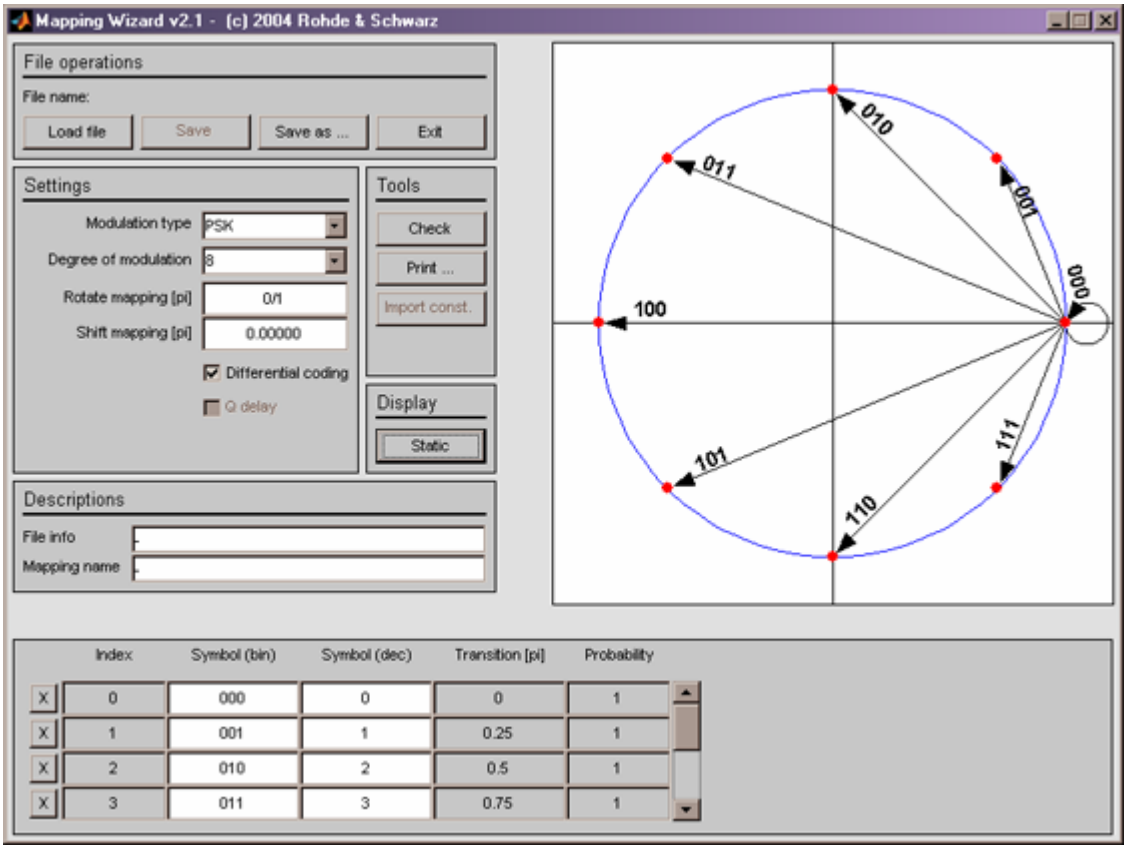
Setting the burst structure

All areas that are marked in the BURST tag generate RF output. The slopes (rise and fall) can be configured with separate burst settings (user interface or remote control). Areas of attenuated RF can be defined within a burst. The CWMODE tag finally defines areas without data content. In fact during CWMODE data continues to play back and the user must take care to fill these areas with dummy data.

Custom Mappings

As mentioned before user defined mappings can be uploaded to the instrument as a binary *.vam file.

A Matlab program called Mapping Wizard (MapWiz) is available free of charge from the Rohde & Schwarz website. This program allows to generate or modify IQ configurations and export them as *.vam file.



Screen shot of Mapping Wizard

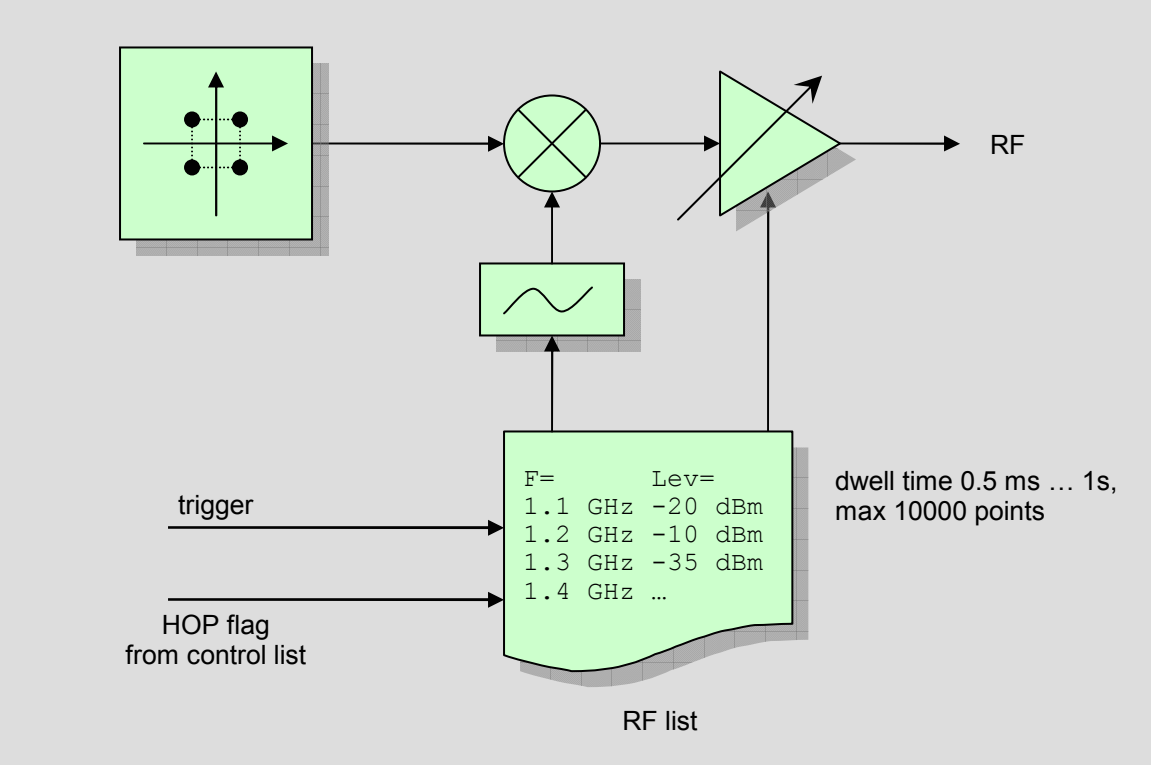
All *.vam files are in binary format and can not be read nor be edited using a regular text editor.

More detailed information on how to use the mapping editor can be found in the MapWiz user manual that is bundled with the application.

If MatLab is not available the Matlab Common Runtime (MCR) can be used to run the application. Please visit the Mathworks homepage (http://www.mathworks.com) for further details.

RF List Mode

The RF list mode can be used to play back a series of previously defined frequencies and levels. In contrast to regular FM or AM sweeps the data pairs are discrete frequency and level values which can be freely defined. List entries do only affect the RF up conversion and thus, frequency lists can be used with pure CW signals as well as any modulated signal from the internal base band generator or the IQ inputs.



Block diagram of RF list mode

Frequency lists can be edited or created with the internal list editor. In addition it is possible to create lists and add data points via remote control commands. More details on how to create the list data on the instrument can be found under the GPIB section of this application note.

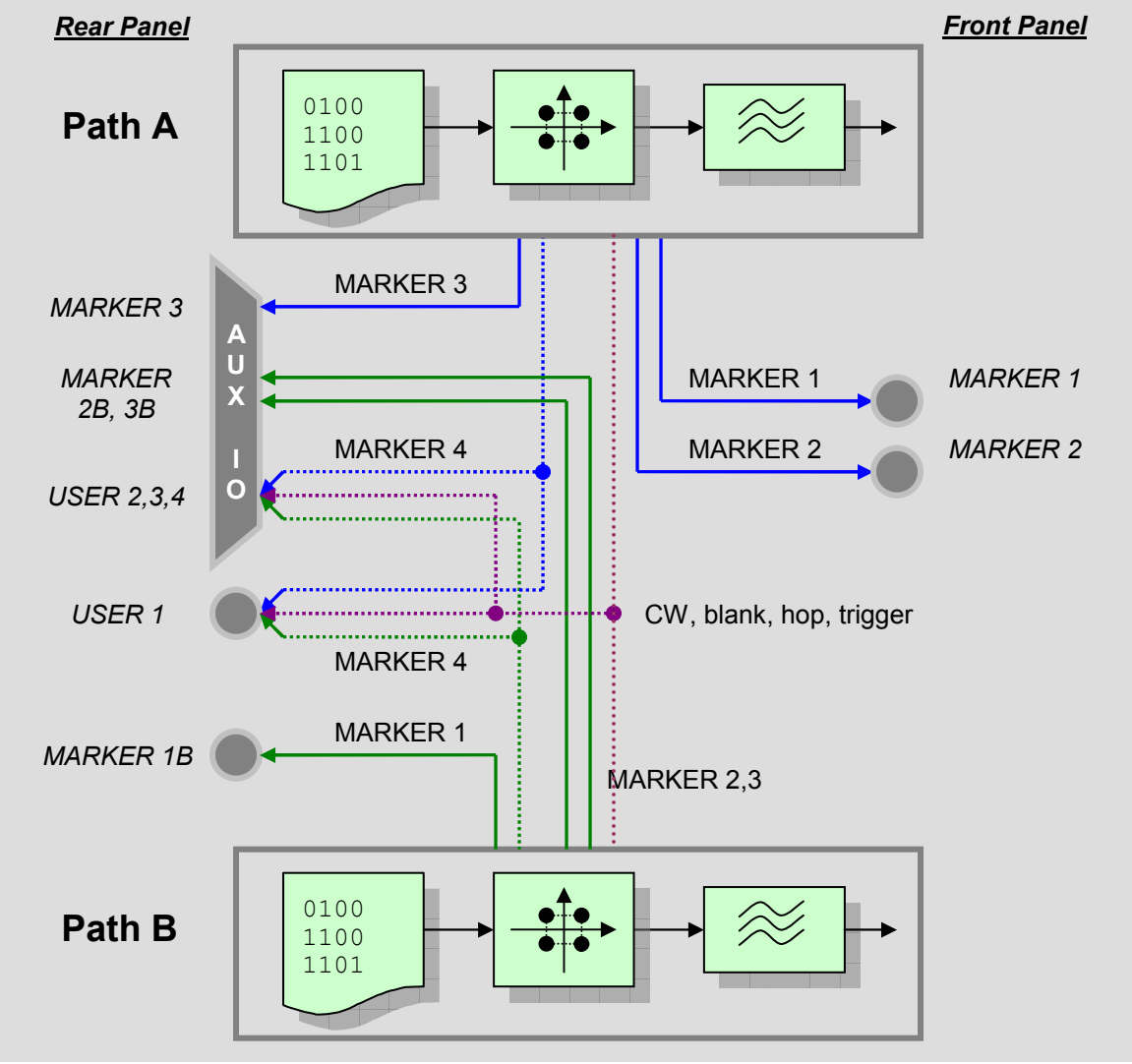
Frequency lists can be played back in continuous mode (cycles) or as a single shot. In the latter case playback is started by a trigger command. Additionally, it is possible to process RF lists in single step mode where the next item is activated by a trigger command or the hop flag in a control list.

List data is usually pre processed on the instrument and is therefore in binary format.

Using Markers

The instruments SMU, SMJ and SMATE offer very versatile marker and trigger configuration options. These signals can be used to trigger external instruments as well as for the synchronization of multiple basebands.

The following picture illustrates the marker routing options.



Block diagram of marker routing

The marker signals 1, 2 and 3 of either path are permanently assigned to their outputs and cannot be changed.

Marker 4 can be routed to any of the user outputs.

Alternatively, other signals than marker 4 can be routed to the user outputs. These signals are the 'CW mode' flag, the blank flag, the hop flag or a trigger. Most of these flags are defined in a control list in custom digital modulation mode.

The flags and the trigger are briefly described below.

CW Mode Flag	Defined in control list. Becomes active during sections of signal playback without modulation (pure CW).
Blank Flag	Generated by Instrument. Becomes active during periods where no output signal is prevalent, e.g. during frequency or level changes in RF list mode.
Hop Flag	Defined in control list. The flag indicates a point where a frequency change should occur, e.g. the next entry in the RF list is processed.
Trigger	The signal is set by either an internally generated trigger event or an external signal.

4 Description of File Formats

As mentioned earlier different file formats exists for the various modes of operation.

Mode	Files	Extension	Format
ARB	Wave Form Files	*.wv	Binary+Text
Custom Dig Mod	Data Lists	*.dm_iqd	Binary+Text
	Control Lists	*.dm_iqc	Text
	Custom Mappings	*.vam	Binary
RF List Mode	RF List Files	*.lsw	Binary

There are three different file types that are dealt with in this application note. One is the waveform file which is used in arbitrary mode and contains raw I and Q values. Second, there is a file called data list which contains binary data that is used as data source with any type of modulation. Last is the control list that can be used to define burst areas and activate marker signals during signal playback.

Common to all files is a simple tag structure that has to be followed. A tag can be regarded as one piece of information which consists of a name section and a data section. The name is used to identify the type of information that is contained in the data section. Data can be in ASCII or in binary format depending on the actual type of information.

The general syntax of a tag is as follows:

{<NAME>: <Data>}

The entire tag is enclosed in curly brackets with the tag name expressed in capital letters. The colon needs to follow the tag name immediately without a blank. After the colon a blank is permissible before the actual data content starts. Tags can follow each other with or without separating blanks or linefeeds.

The foregoing section provides an overview of all available tags for the various file formats. All of the listed tags are shown as they could be used in a real file and are not displayed as syntax description. Areas that contain data and may be changed are underlined.

Waveform Files

Waveform files contain ASCII text as well as binary data. It is possible to create these files on a PCs local hard drive and upload them to the instrument. However, the data tag contains binary data and therefore it is not recommended to view or edit these files with simple text editors such as Microsoft Notepad.

The following overview shows all tags that can be used in a waveform file. There is no specific order in which these tags have to be arranged, with two exceptions. The type tag must be the very first one and the waveform data tag must always be the final one.

{TYPE: SMU-WV,0}	Mandatory. This tag sets the file format to be a wave form. SMU-WV is also used for the SMATE and SMJ. The 0 indicates that no checksum is used.
{CLOCK: 54000000}	Mandatory. Plays back the waveform at 54 MHz. The maximum is 100 MHz clock rate. Clock rate in Hz.
{COMMENT: text}	Descriptive text that is shown when the waveform is selected.
{COPYRIGHT: text}	Displays a copyright note.
{DATE: 2005-11-24;17:55:00}	Specifies the waveform creation date.
{CONTROL LENGTH: 5}	Set the length of the embedded control list in bytes.
{MARKER LIST 1: 0:1;1:0}	Sets marker 1 to HIGH on the first sample and LOW from the second on. There are 4 marker lists available.
{LEVEL OFFS: 3.0,1.0}	Mandatory. Level offset. RMS Offset is first, Peak Offset is second value. Values in dB fully scale.
{SAMPLES: 5}	The waveform contains 5 IQ data pairs.
{WAVEFORM-21: #iqiqiq…}	Mandatory. Contains the actual waveform data. 21 bytes = 20 IQ data bytes plus the leading hash character. This tag must be the last one in the file. I and Q data is 16 bits each, LSB first.

```
{TYPE: SMU-WV,0}
{COMMENT: I/Q sine wave, f=100.000000kHz, IQPairs=512}
{DATE: 2005-11-25;12:33:51}
{CONTROL LENGTH: 512}
{MARKER LIST 1: 0:1;1:0}
{CLOCK: 51200000}
{LEVEL OFFSET: 0.000000,0.000000}
{SAMPLES: 512}{WAVEFORM-2049: #...}
```

The example file contains 2048 data bytes (512 IQ samples, I and Q 16 bit each) which are replaced by three dots […]. Including the mandatory leading hash character this results in a total

byte count of 2049. The use of the 'CONTROL LENGTH' tag is mandatory if any marker list tags are used. This tells the instrument how long the associated control list data is and ensures that markers and data remain in sync. For more information about control list tags refer to the control list section in this document.

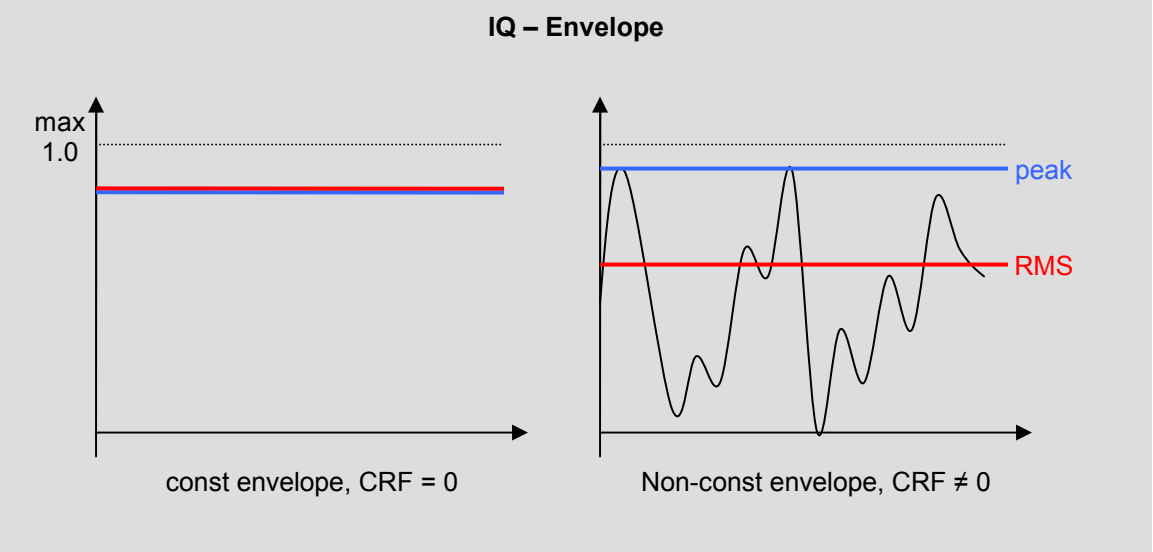
The level offset tag requires a more detailed explanation. It contains two numbers, one for the RMS offset in dB_{fs} (dB full scale) and a second one for the peak offset in dB_{fs}.

The RMS, Peak and Crest Factor of a complex signal is defined as follows:

$$RMS = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1}(I_n^{\,2} + Q_n^{\,2})} \quad , \quad Peak = Max\left\{\sqrt{I_n^{\,2} + Q_n^{\,2}}\right\} \quad , \quad Crest = 20 \cdot \log_{10}\left(\frac{Peak}{RMS}\right)$$

For complex signals with a constant envelope (vector length) the peak and RMS value is identical. In this case the crest factor becomes zero. If the envelope is exactly 1.0, which means that the full signal range is used, both numbers of the level tag need to be set to zero. However, it is possible to use other values that will either scale down the signal or lead to clipping.

Both situations might be desirable under certain circumstances. An example could be a signal that generally uses low levels. Once in a while high peaks exist where a clipping situation can be tolerated. Under these circumstances a negative value for the peak offset will scale up the signal and result in a higher resolution of the low level signal sections.



IQ Envelopes

Both numbers in the level offset tag are specified in dB – full scale. Positive numbers decrease the level, whereas negative numbers increase the level and may lead to clipping.

$$Peak\,/\,dB_{fs} = 20 \cdot \log_{10}\left(\frac{Full}{Peak}\right) \quad , \quad RMS\,/\,dB_{fs} = 20 \cdot \log_{10}\left(\frac{Full}{RMS}\right)$$

If the signal full scale is 1.0 and the RMS of the waveform was calculated to 0.707 the RMS Offset would be 3 dB.

Data List Files

Data lists are similar structured to waveform files. The same rules apply to the type tag which needs to be the very first one in the file as well as the data tag which needs to be the very last one. The following overview lists all important tags for this file format.

{TYPE: SMU-DL,0}	Mandatory. Set the file format to data list. The checksum is always zero for data lists. SMU-DL is also used for the SMATE and SMJ.
{COMMENT: text}	Descriptive text that is shown when the data list is selected.
{COPYRIGHT: text}	Displays a copyright note.
{DATE: 2005-11-24;17:55:00}	Specifies the data list creation date.
{DATA BITLENGTH: 444}	Mandatory. Set the bit length of the entire data list.
{DATA LIST-56: #bbbbb…}	Mandatory. Actual data list. 8bit, MSB first. Length is in full bytes. Format is binary. This tag must be the last one in the file.

```
{TYPE: SMU-DL,0}
{COMMENT: Data List, 600 Symbols, QPSK}
{DATE: 2005-11-28;16:31:09}
{DATA BITLENGTH: 1200}
{DATA LIST-151: #...}
```

The example transfers 150 data bytes which are replaced by three dots […].

In the example 1200 data bits are to be transferred. Since each byte consists of 8 bits there are 150 data bytes required. Including the leading hash (which is mandatory) the resulting data list size is 151 characters.

The reason for specifying the total bit length separately is that there could be a number of bits that does not exactly match up a full byte count. In this case the last remaining byte has to be filled with dummy bits and the bit count specifies how much data is actually valid.

All data is specified in eight bit format, MSB first.

It should be noted that the actual data area is pure binary and thus, it is not human readable. This needs to be considered when editing these files with regular text editors as they may convert non-ASCII characters or add extra linefeeds.

Control List Files

Control lists use the same tag structure as waveforms and data lists but in contrast to the other files they are text only. It is therefore possible to edit control files using a regular text editor but care must be taken that no extra line feeds are added within tags.

As in the other files, the type tag must be the very first one to tell the parser what file type to analyze. The following overview lists the available tags.

{TYPE: SMU-CL,0}	Mandatory. Set the file format to control list. The checksum is always zero for control lists. SMU-CL is also used for the SMATE and SMJ
{COMMENT: text}	Descriptive text that is shown when the control list is selected.
{COPYRIGHT: text}	Displays a copyright note.
{DATE: 2005-11-24;17:55:00}	Specifies the control list creation date.
{CONTROL LENGTH: 200}	Mandatory. Set the symbol length of the entire control list.
{MARKER LIST 1: 0:1;1:0}	Sets marker 1 to HIGH on the first sample and LOW from the second on. There are 4 marker lists.
{BURST LIST: 0:1;200:0}	Set burst state if power ramping is active.
{LEVATT LIST 1: 0:1;200:0}	Define level attenuation area if power ramping is active.
{CW MODE LIST: 0:1;200:0}	Define areas without modulation.
{HOP LIST: 0:0;50:1;51:0}	Set a point for frequency hop.

```
{TYPE: SMU-CL,0}
{COMMENT: Control List, 600 Symbols, QPSK}
{DATE: 2005-11-28;16:31:09}
{CONTROL LENGTH: 600}
{MARKER LIST 1: 0:1;1:0}{MARKER LIST 2: 0:0}
{MARKER LIST 3: 0:0}{MARKER LIST 4: 0:0}
{BURST LIST: 0:1;200:0;400:1}
{LEVATT LIST 1: 0:0;400:1}
{CW MODE LIST: 0:0}{HOP LIST: 0:0}
```

The example shows a control list that is used with 600 symbols. Marker signal 1 is set to high for only the very first symbol in the data stream. In addition the burst tag specifies that only the symbols between 0-199 and 400-599 generate RF output. The symbols 200-399 are blanked. The LEVATT list sets the symbols 400-599 to an attenuated state (lower RF power).

RF List Files

RF List files are stored in binary format. It is recommended to not directly create these files outside the instrument but rather use remote control commands to assemble them on the instrument. The reason is that instrument specific data is added once the list is complete which in turn speeds up frequency hopping. The addition of instrument specific data is referred to as a learning process and can be compared to compiling the source code of a program. A typical sequence would be as follows:

- Setup baseband signal
- Ensure that RF list mode is not already active
- Set new list name
- Add data points to the list
- Learn list data
- Setup parameters, such as dwell time and playback mode
- Start playback

Typical frequency settling times are below 450 µs if the RF level remains constant. During the frequency or level hop the signal is blanked for a short period of time to avoid over shots. This blanking period activates a blank signal which can be routed to one of the user outputs.

Please refer to the GPIB section in this document to see how lists are created and data is added.

5 Converting and Uploading Existing Data

This paragraph describes how existing data can be transferred to the instrument using already available software. In addition, some advice is provided for own implementations. This section is divided into several sub sections that deal with different data and situations.

The '**ASCII Lists**' section describes how text data can be transferred to the Instrument using IQWizard.

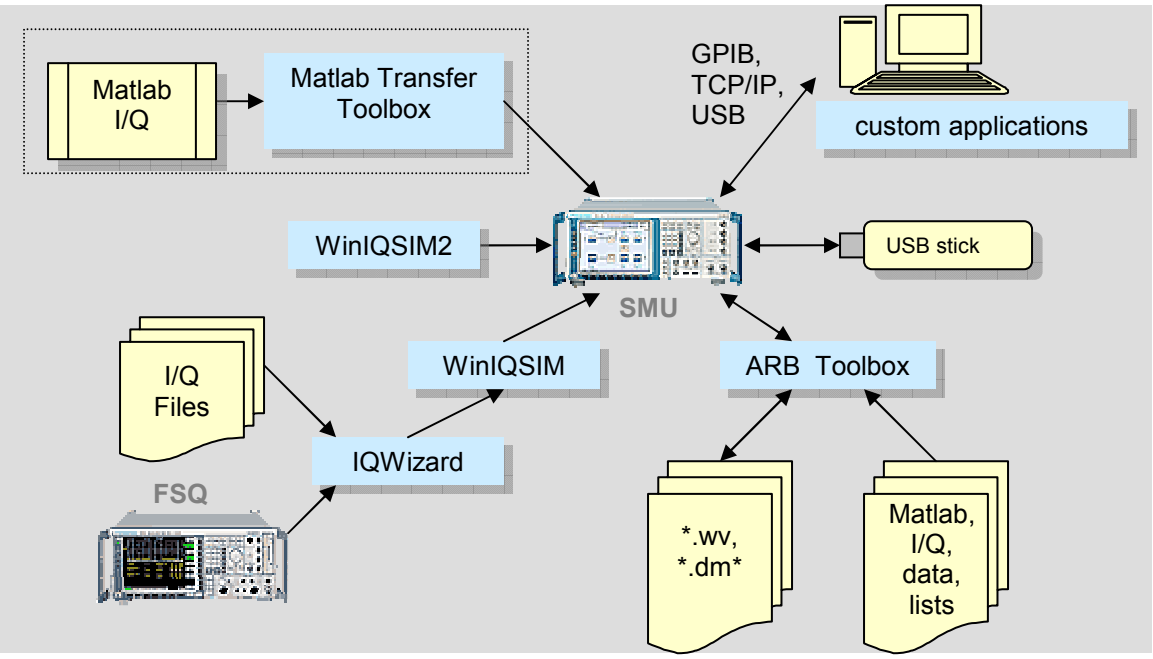
The '**Matlab Data**' section describes a tool box that is available from the Rohde & Schwarz web site and provides functions for connecting to an instrument and uploading data.

The '**Generating Waveforms using WinIQSIM**' paragraph shows how this software can be used to generate custom data and upload it to an instrument. In addition WinIQSIM can interface with IQWizard to provide a powerful import filter for various formats. Both programs are available free of charge from the Rohde & Schwarz web site.

If only FM/AM or phase modulated signals are required the section '**FM/AM/PM**' introduces a software package that can generate and upload these waveforms.

The following paragraph '**Using Captured Data from a Spectrum Analyzer**' shows how IQWizard can be used to transfer captured data to WinIQSIM for upload.

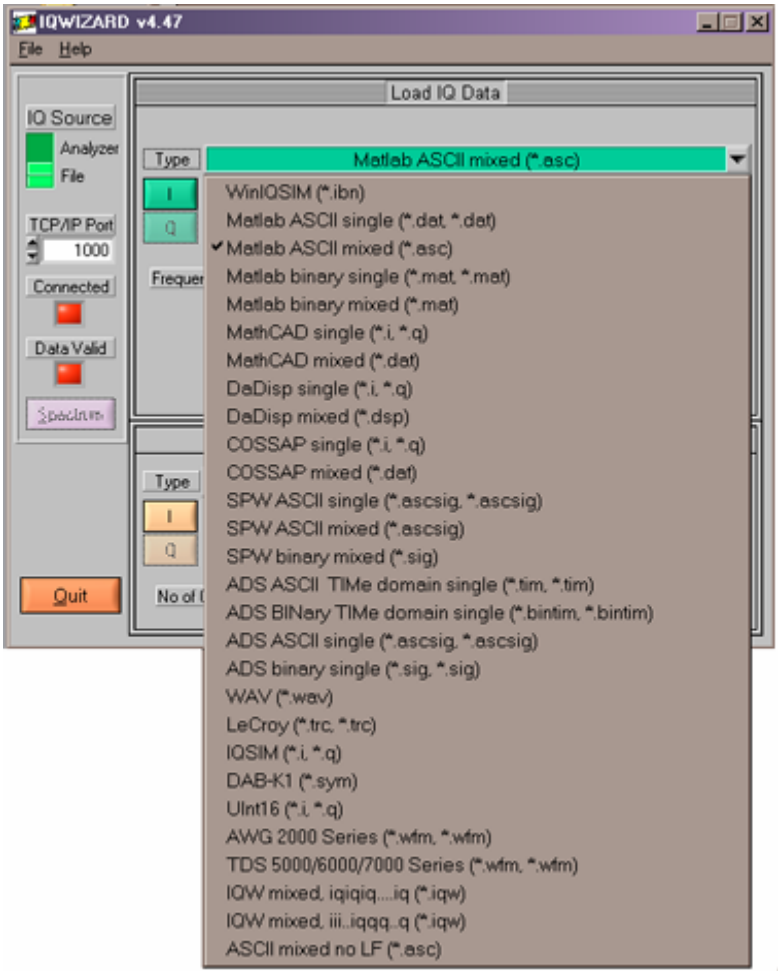
The last two sections '**Useful GPIB Commands**' and '**Communicating using VISA**' are starting points for own application development.



Exchanging data with the instrument

ASCII Lists

Data in ASCII format can be imported with IQWizard [2]. This tool is available free of charge from the Rohde & Schwarz web site and provides a large range of import filters.



Screen shot of IQWizard

The format of choice is 'Matlab ASCII mixed' if all data is present as two columns of ASCII formatted real numbers.

If data is formatted as space separated values (alternating I and Q) 'ASCII no LF' should be used.

A full overview with an explanation can be found in the application note for IQWizard. Please refer to the reference section of this application note for information about where to obtain a copy.

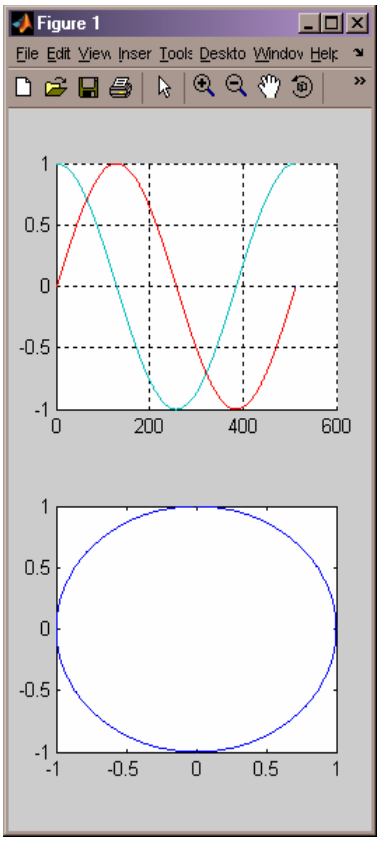
Matlab Data

For Matlab users a separate application note (1GP60) exists that describes the Matlab transfer toolbox [3] which is available free of charge from the Rohde & Schwarz web site. Please see the reference section of this document for further details about where to obtain a copy of the tool box.

The transfer tool box provides functions for instrument connection and upload of arbitrary data as well as processing SCPI batch files. The following table shows a brief overview of the functionality.

Function	Use
`rs_connect`	Create the device object
`rs_send_command`	Send a GPIB command
`rs_send_query`	Send a GPIB command and return the answer
`rs_batch_interpret`	Run a sequence of commands from a simple text file
`rs_generate_wave`	Upload waveform data to the instrument

All of the above functions are provided as source code (.m files) and can be changed as needed.



The code example section of this application note contains an example Matlab script that demonstrates the use of the Matlab Tool Box.

The script contains one single function that is called rs_demo and does the following:

- Connect to Instrument
- Read identification string
- Reset Instrument
- Apply RF settings
- Calculate IQ Waveforms
- Display Waveforms
- Upload Data to Instrument
- Start Playback

If the tool box is not an option the tool IQWizard [2] is capable of converting Matlab data into other file formats or interface directly with WinIQSIM. IQWizard is available free of charge from the Rohde & Schwarz web site. An application note is provided with the software that describes the functionality in more detail. For interfacing with WinIQSIM please refer to the WinIQSIM section of this document. The following table gives a brief overview of the supported Matlab related data formats.

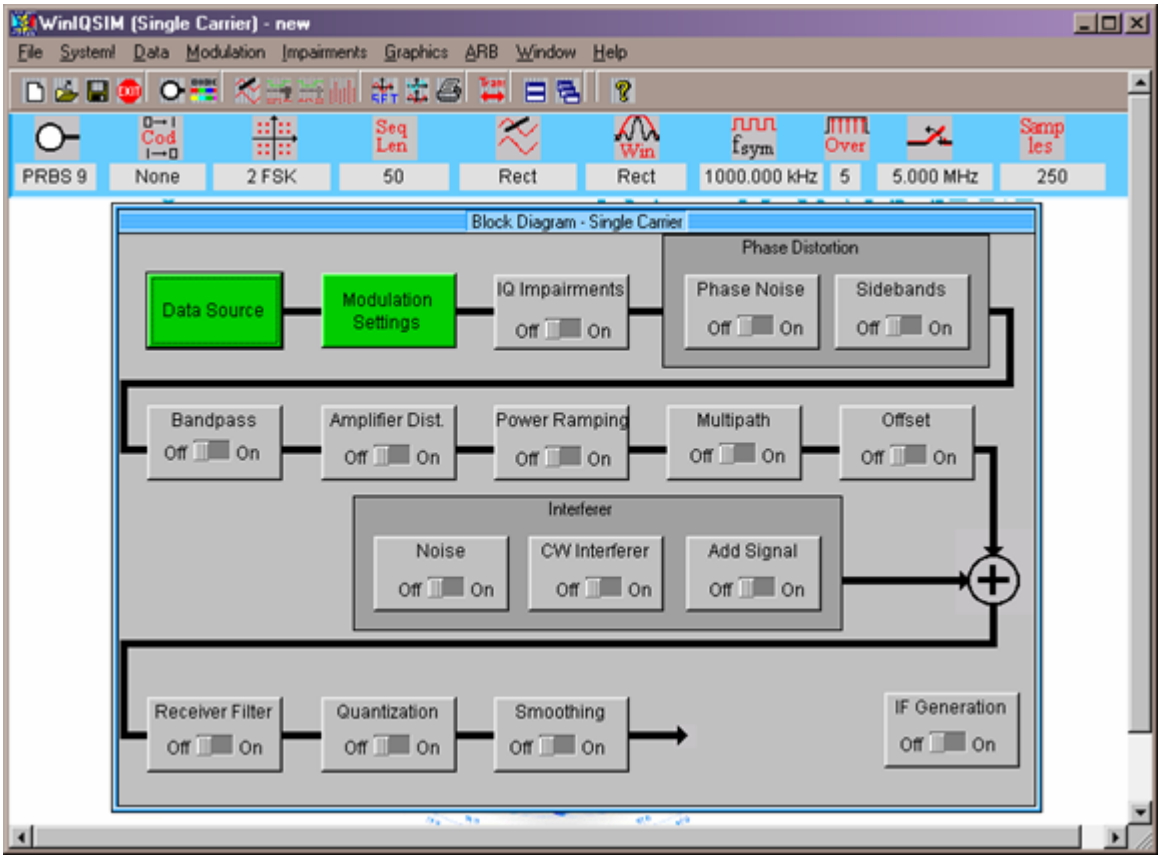
File	Type
`*.dat , *.dat`	ASCII single
`*.asc`	ASCII mixed
`*.mat , *.mat`	Binary single
`*.mat`	Binary mixed

The file format of the *.dat and *.asc files are quite simple. The files contain either one or two columns of numbers that represent the I and Q values:

```
1.000   0.200
1.100   0.250
1.250   0.310
   …
```

Generating Waveforms with R&S WinIQSIM™

WinIQSIM [5] is a very powerful application that generates arbitrary waveform data for many standards. In addition it allows to add noise, fading, distortion and filter settings. WinIQSIM is available free of charge from the Rohde & Schwarz website and comes with its own detailed manual.



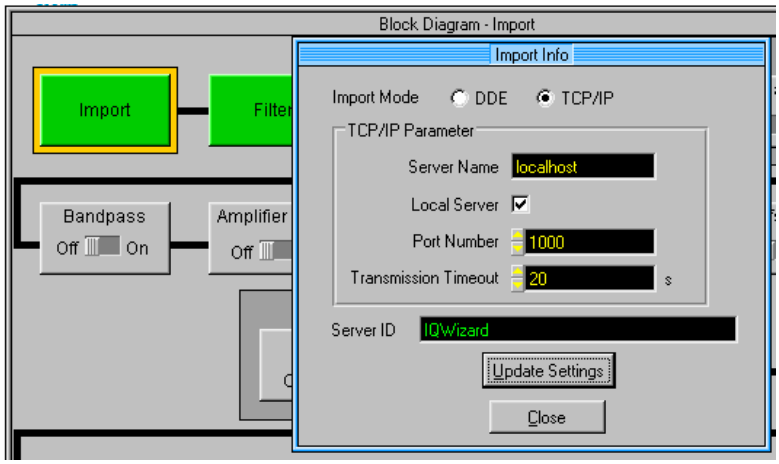
Screen shot of WinIQSIM

WinIQSIM interfaces directly with instruments such as the SMU, SMJ or SMATE. Waveforms are transferred to the instrument and can also be saved to the local hard drive. If waveforms for standards such as 3GPP, CDMA2000 or WMAN are to be generated an additional option key may need to be installed on the instrument.

WinIQSIM can interface directly with IQWizard through a TCP/IP link. This mechanism can be used to convert existing waveforms and directly upload them to the instrument. WinIQSIM receives the wave data as imported waveform and can apply all its modifications (e.g. noise) to the received data.

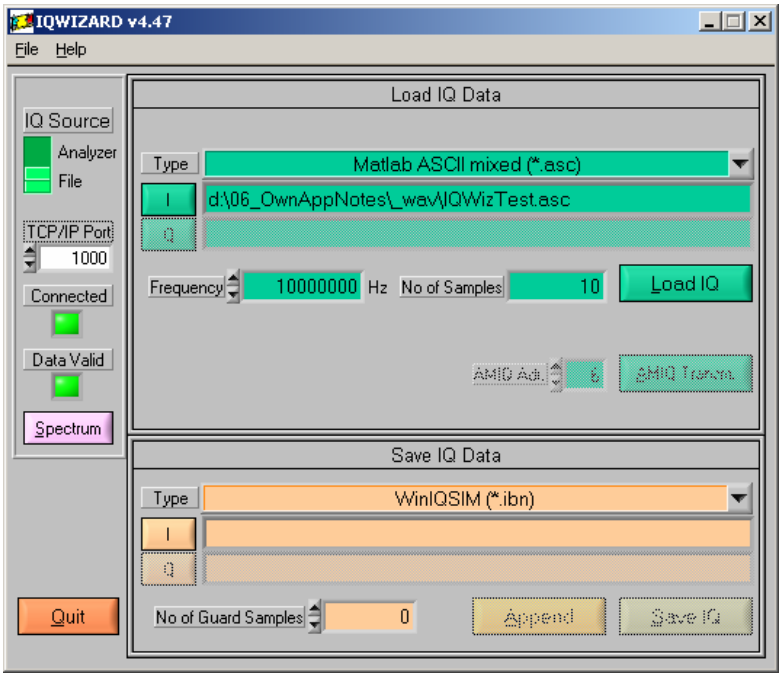
Connection Example

- Start WinIQSIM and select System from the menu bar. In the system selection window select 'Import'. Now click on the green 'Import' button in the block diagram which should bring up the import settings dialog. Make sure to use an available TCP/IP port number and hit 'Update Settings'.



Screen shot of import dialog

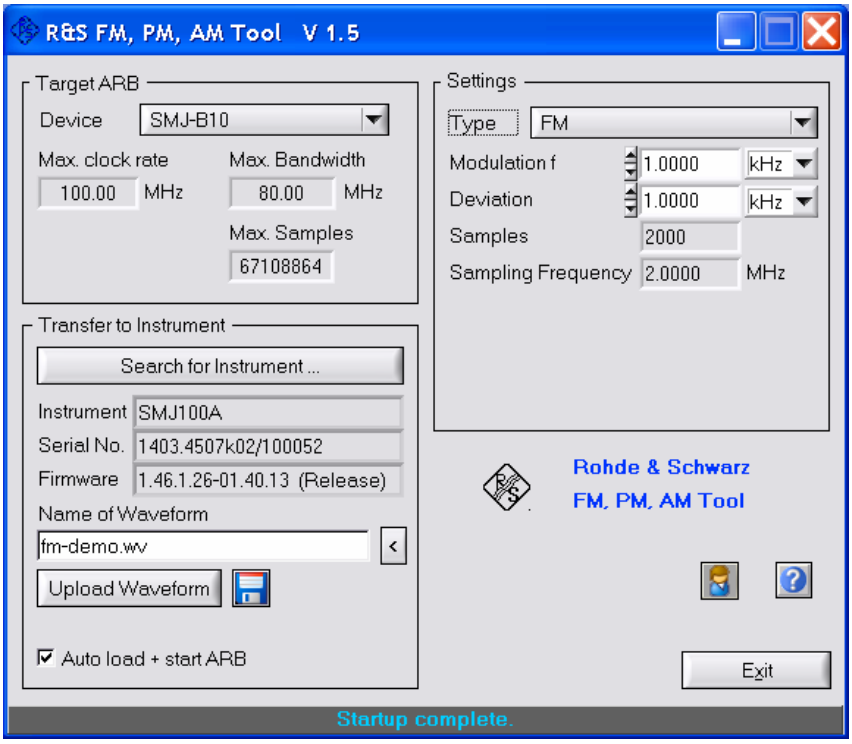
- Next bring up IQWizard and setup the TCP/IP port to the same as above. Initially the connection LED will be red and turn to green as soon as the link is up. If the LED remains red try using different TCP/IP ports.



Screen shot of IQWizard

FM/AM/PM with R&S FM/AM ARB software

The FM/AM tool [1] can be used to generate arbitrary waveforms for FM, AM and PM modulation. The software is available free of charge from the Rohde & Schwarz website.



FM/AM ARB Tool

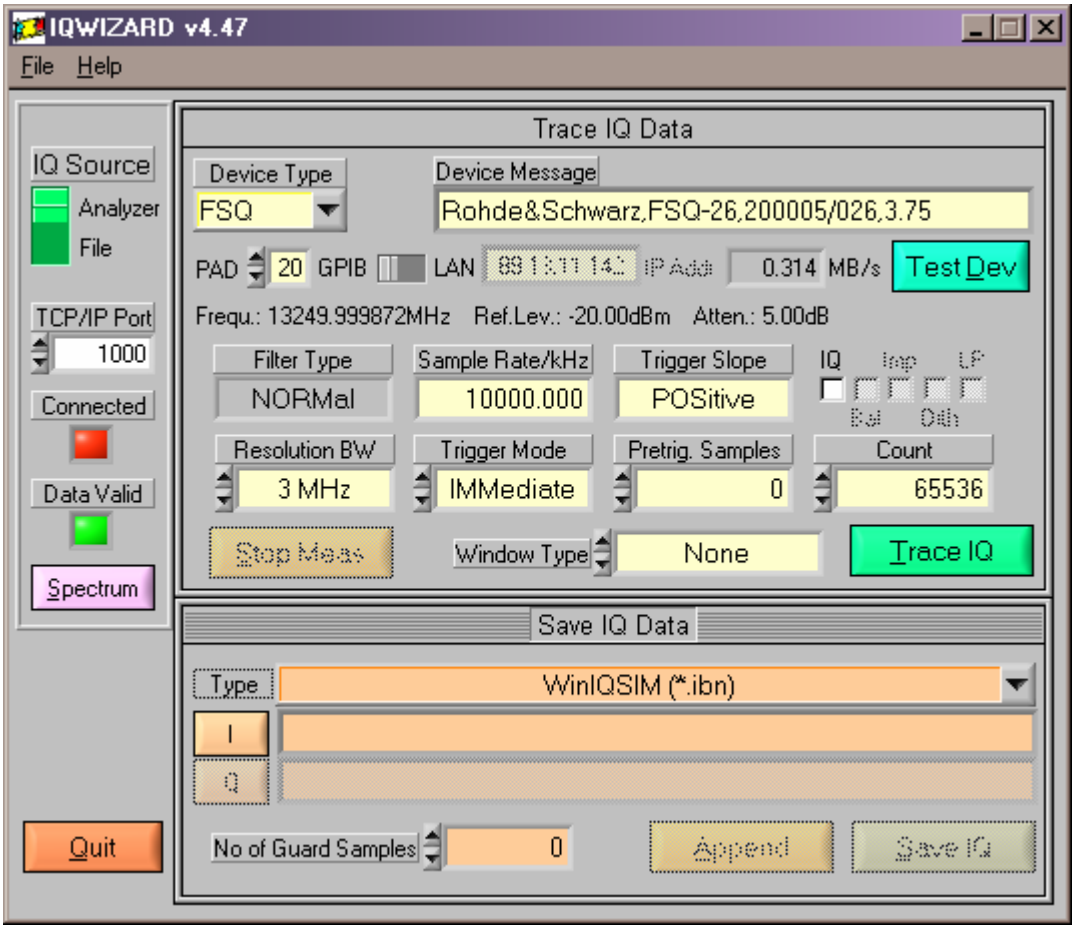
The picture shows the R&S FM/AM ARB tool.

1. Select the correct target instrument under ‘Target ARB’.
2. Select the type of modulation.
3. Select your desired modulation frequency as well as the deviation.
4. Click on ‘Look for instrument…’.
5. Available instruments are shown with a green light. Click on the green light and then ‘Select Instrument’
6. Type in a name of the waveform. This is ‘FM’ in the example.
7. Check ‘Auto load and start waveform’. The path selection box will only be available for dual base band signal generators.
8. Click ‘Transfer File’ which will popup a dialog that shows all waveforms that are stored on the instrument.

More detailed information and a user manual can be found in the application note for R&S FM/AM ARB [1]. Please see the reference section of this document for further details.

Using Captured Data from a Spectrum Analyzer

If IQ data is to be captured and used as arbitrary data for the signal generator the tool IQWizard may be used. IQWizard and its application note is available free of charge from the Rohde & Schwarz web site.



Screen shot of IQWizard

IQWizard currently supports the instruments FSQ, FSP, FSU, FSIQ, FSL and ESPI and allows to store data in various file formats.

If the captured data is saved as *.ibn file it can be loaded and transferred to the instrument using WinIQSIM. Alternatively IQWizard can directly interface with WinIQSIM via TCP/IP link to transfer its data.

Useful GPIB Commands

This paragraph briefly summarizes important commands that are used for the various operating modes. All commands are shown as an example to make their use more obvious. A full command reference, however, can be found in the programming section of the instrument. The information of this chapter is useful for people who desire to write their own applications. Some advice is given on how to use the remote control commands and how to avoid common mistakes.

The following is a list of general mass memory commands. These commands can be used to upload files – binary or text to almost any instrument location. The two commands that control the line termination are required whenever binary data must be transferred. This is particularly important to avoid the misinterpretation of characters as end of transmission flags that would disruption the transmission.

Code examples are included in the code example section of this application note.

`:MMEM:MSIS 'D:'`	Set the default drive to D:
`:MMEM:CDIR '\'`	Set the default directory to \
`:MMEM:DEL 'D:\demo.wv'`	Delete a file
`:MMEM:DATA 'D:\demo.wv',#3151<binary>`	Transfers binary data to the instrument
`:SYST:COMM:GPIB:LTER EOI`	Set the instrument to use the EOI line as additional line terminator.
`:SYST:COMM:GPIB:LTER STAN`	Set the instrument back to standard termination (line feed).

The format of :MMEM:DATA requires some syntax rules to be followed:

`:MMEM:DATA '<filename>',#<len of data_len><data_len><data bytes>`

<filename> The path and file name of the file that is to be generated. The file can be specified with an absolute path, e.g. D:\demo.wv to ensure storage at a certain location. If no absolute path is used the default location is used.

<len of data_len> This is the number of digits that immediately follow and are used to specify the actual data length. It can be calculated from the data length as follows:
`(int) floor( log10(data_len) ) + 1`

<data_len> The actual length of the data block in bytes as ASCII formatted number.

<data bytes> Binary data.

The example above '#3151<binary>' transfers 151 data bytes. Since 151 is a three digit number (human readable ASCII) its length is 3.

MMEM:MSIS and MMEM:CDIR should always be used and the default path be set to D:\. This ensures that user data is not accidentally erased while performing a software upgrade or data recovery on the instruments C: drive.

In addition to the general mass memory commands dedicated command groups exist for the according operation mode, e.g. arbitrary, data list playback or RF list playback.

Waveform related commands

This section provides an overview of the relevant commands for waveform playback with arbitrary IQ data. In order to ensure a certain file location on the file system it is recommended to use an absolute path name, e.g. 'D:\'.

`BB:ARB:WAV:CAT?`	list all waveform files in default path (*.wv)
`BB:ARB:WAV:DATA 'D:\demo.wv',#3151<binary>`	copy binary data to waveform file
`:SOUR1:BB:ARB:WAV:SEL 'D:\demo.wv'`	select arbitrary waveform for path A
`:SOUR1:BB:ARB:STAT ON`	activate ARB generator for path A

BB:ARB:WAV:DATA can be used in lieu of MMEM:DATA. However, there is an important difference between these commands. The mass memory functions always replace existing files entirely whereas the list mode related commands (e.g. BB:ARB…) overwrite files starting at position zero. Thus, using BB:ARB only increases the file size.

The prefix :SOUR1 or :SOUR2 can be omitted if only one base band channel exists, e.g. the SMJ 100A.

Data and Control Lists

For data and control lists a similar set of commands exist. Using a control list is optional and only required for markers, attenuated or blanked areas as well as pure CW sections.

`BB:DM:DLIS:CAT?`	list all data lists in default path (*.dm_iqd)
`BB:DM:DLIS:DEL 'D:\demo'`	remove a data list
`:SOUR1:BB:DM:DLIS:SEL 'D:\demo'`	select data list for path A

`BB:DM:CLIS:CAT?`	list all control lists in default path (*.dm_iqc)
`BB:DM:CLIS:DEL 'D:\demo'`	remove a control list
`:SOUR1:BB:DM:CLIS:SEL 'D:\demo'`	select control list for path A

No specific commands exist to directly copy data and control lists to the instrument. Instead the mass memory functions must be used.

The prefix :SOUR1 or :SOUR2 can be omitted if only one base band channel exists, e.g. an SMJ 100A.

Data List Related Commands

The data and control lists do not contain any information related to the modulation, burst shape and filter settings. Thus, additional commands exist to control these paraameters.

`SOUR1:BB:DM:SOUR DLIS`	activate data list mode
`SOUR1:BB:DM:FORM QPSK`	select modulation
`SOUR1:BB:DM:FILT:TYPE RCOS`	select root COS filter
`SOUR1:BB:DM:FILT:PAR:RCOS 0.35`	set filter roll off
`SOUR1:BB:DM:SRAT 100.0 kHz`	set symbol rate
`SOUR1:BB:DM:PRAM ON`	activate power ramping
`SOUR1:BB:DM:PRAM:ATT 15.0 dB`	set level attenuation
`SOUR1:BB:DM:TRIG:OUTP1:MODE CLIS`	set data source to CLIST for marker 1
`SOUR1:BB:DM:STAT ON`	activate digital modulation

This set of commands is intended as an overview. Particularly for the various filter settings a large amount of additional commands exist. For a detailed overview please refer to the programming section of the instruments operating manual.

The prefix :SOUR1 or :SOUR2 can be omitted if only one base band channel exists, e.g. an SMJ 100A.

It is important to activate power ramping by the use of BB:DM:PRAM ON whenever bursts or attenuated signal areas are required. The other way around signal sections that should generate RF output must explicitly be included in the burst area once power ramping is active.

In other words, the BURST LIST tag is optional without power ramping active but it becomes mandatory with power ramping enabled. Besides using the control list to define power ramping it is also possible to use an external signal.

RF List Mode

The following table shows the most important commands for use with the RF list mode. As mentioned before the actual RF list contains instrument specific data. Therefore it is not recommended to upload lists that were created outside of the instrument. Lists should rather be created on the actual instrument using the following remote control commands.

`SOUR:LIST:CAT?`	Lists all RF list files (*.lsw)
`SOUR:LIST:DEL 'demo'`	Removes RF list file
`SOUR1:FREQ:MODE CW`	Deactivate list mode
`SOUR1:LIST:SEL 'D:\demo'`	Selects or creates a list for path A
`SOUR1:LIST:FREQ 1.0 GHz, 1.1 GHz`	Fills selected list with frequencies
`SOUR1:LIST:POW -20 dBm, -22 dBm`	Fills selected list with power values
`SOUR1:LIST:DWEL 1 ms`	Set dwell time for list playback
`SOUR1:LIST:MODE AUTO` `SOUR1:LIST:MODE SING`	Continuous playback Single playback, started via LIST:TRIG:EXEC
`SOUR1:LIST:TRIG:SOUR AUTO`	Set trigger mode to auto
`SOUR1:LIST:LEAR`	Learn list settings
`OUTP1:STAT ON`	Switch RF signal on
`SOUR1:FREQ:MODE LIST`	Activates the list mode for path A
`LIST:TRIG:EXEC`	Start list playback (trigger)

The above table lists the GPIB commands in the order as they should be used.

1. An existing list should be deleted using the MMEM:DEL or LIST:DEL command.
2. The mode is set back to CW in order to turn list mode off. If IQ modulation was used before it is also suggested to switch this mode off as well and going back to pure CW.
3. The new list file is created with the LIST:SEL command.
4. Data points are added to the list using LIST:FREQ and LIST:POW.
5. The remaining settings should be made. This is usually the dwell time and the trigger settings.
6. Most important the list is ‘learned’ by the LIST:LEARN command. This processes the newly created list and adds instrument specific information to speed up frequency hopping.
7. STAT:ON must be called to activate the RF output before the actual list mode is activated.
8. Finally the list playback is activated via FREQ:MODE LIST.
9. Depending on trigger settings it may be required to manually start playback by issuing the TRIG:EXEC command. This is only required in single trigger mode.

A more detailed explanation and a list of all available commands can be found in the instrument operating manual.

Communicating using VISA

VISA stands for Virtual Instrument Software Architecture. The general idea behind VISA is to provide an abstraction layer for instrument control. Thus, VISA contains a very large collection of functions to scan for instruments, open and close sessions as well as transferring data. An additional benefit is that programs written for VISA will most likely not depend on the actual hardware link to the instrument. It is possible to use VISA with traditional GPIB cards as well as TCP/IP to GPIB converters or straight TCP/IP (VXI-11) links.

VISA can be obtained from multiple companies but the use of the National Instruments hardware and software is suggested in this application note. Further details can be found on the National Instruments web site.

The following sections lists some useful C functions that are required when remote controlling the instrument with VISA.

```
Open instrument connection
      ViSession      rmSession, instrSesssion;
      ViChar         sRscName[] = "GPIB0::28::INSTR";
      viOpenDefaultRM (&rmSession);
      viOpen (rmSession, sRscName, VI_NULL, VI_NULL, &instrSession);
Connection and timeout settings
      viSetAttribute (instrSession, VI_ATTR_TMO_VALUE, 5000);
      viSetBuf (instrSession, VI_WRITE_BUF, 2048;
      viSetBuf (instrSession, VI_READ_BUF, 2048);
      viSetAttribute (instrSession, VI_ATTR_WR_BUF_OPER_MODE, VI_FLUSH_ON_ACCESS);
      viSetAttribute (instrSession, VI_ATTR_RD_BUF_OPER_MODE, VI_FLUSH_ON_ACCESS);
Write ASCII data
      ViChar sOutBuf[] = "*RST";
      viPrintf (instrSession, sOutBuf );
Set instrument back to local mode
      viGpibControlREN (instrSession, VI_GPIB_REN_DEASSERT_GTL);
Close connection
      viClose (instrSession);
      viClose (rmSession);
```

The example section of this application note shows how to use these functions in C.

The transfer of large binary data requires to set the VISA interface to use the assertion of END in addition to the end of line character. If this is not done, bytes from the binary data stream may be misinterpreted as end of line and the transmission will be disrupted. The following VISA commands can be used to perform the binary transfer correctly.

```
Require asserted END at end of transmission
     viSetAttribute (instrSession, VI_ATTR_SEND_END_EN, VI_TRUE);
Write Binary Data
     viWrite (instrSession, (ViBuf)cOutBuf, iBufLen, &iSentBytes);
Do not require asserted END anymore
     viSetAttribute (instrSession, VI_ATTR_SEND_END_EN, VI_FALSE);
```

In addition the instrument may require to have its termination mode changed accordingly. This can be done through the following two GPIB commands:

```
:SYST:COMM:GPIB:LTER EOI;
:SYST:COMM:GPIB:LTER STAN;
```

Care should be taken if binary files are written or opened for read using the C function ‘fopen’. Standard settings usually expect text data and cannot be used for binary files. The correct call to the function would therefore be:

```
FILE *fp = fopen ("D:\\data.wv", "rb");
…
fclose (fp);
```

The function opens the file D:\data.wv for read in binary mode.

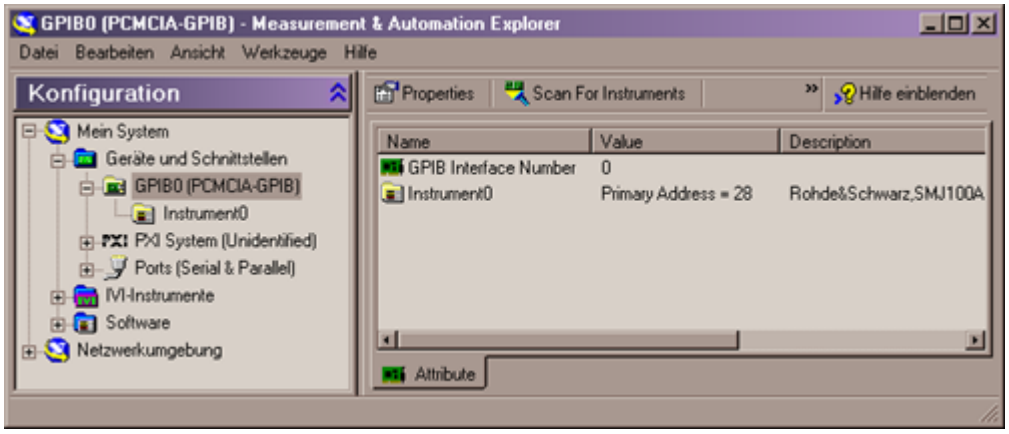
Care must be taken when large amounts of data shall be uploaded. It is possible to split up the transfer into multiple calls to viWrite which would allow progress status reports or a check for user breaks. In addition write buffer settings may restrict the total number of bytes that can be passed on to viWrite. Since the write function usually indicates its end of transfer by asserting EOI this behavior must be switched off for all but the final write operation. This can be achieved by setting the instrument to require EOI and setting VI_ATTR_SEND_END_EN to VI_FALSE. For the final write this attribute is changed to VI_TRUE and the transfer will terminate properly.

6 Transferring Data to the Instrument

GPIB Remote Control

The most common way to interface with test equipment is still GPIB. This section briefly describes the necessary steps to connect the signal generator via this interface.

It is assumed that the reader already has his operational GPIB hardware installed in a PC and the appropriate cable connections are made.



The National Instruments Measurement & Automation Explorer is able to scan the local GPIB bus and verify a good instrument link.

The bus address of the instrument usually defaults to 28 and can be altered in its setup dialog. The following steps describe the procedure for the SMU or SMJ.

- Push the SETUP button on the left side of the front panel.
- Navigate to the GPIB entry of the menu tree and open the dialog.
- Set the desired address and press ENTER.
- Try to scan the bus using tools that are provided with your GPIB hardware (e.g. Measurement & Automation Explorer).

The sequence is similar for the SMATE instrument, except that is needs to be operated with an externally connected USB mouse, keyboard and monitor.

The following C code example shows the call to the 'viOpen' function when used with a local GPIB interface.

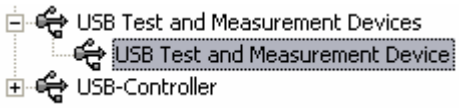
```
viOpen (rmSession, "GPIB0::03::INSTR", VI_NULL, VI_NULL, &instrSession);
```

Important to mention is the resource string that identifies the instrument. GPIB0 references the first GPIB board whereas 03 is the GPIB address of the instrument that is connected to this board.

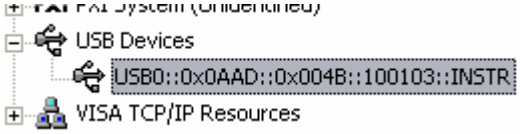
Programming examples can be found in the examples sections of this application note.

USB Remote Control

The AFQ100A offers remote control capabilities via USB connection. If VISA is installed the instrument is automatically detected and added under 'USB Test and Measurement Devices' on the hardware tab of your Microsoft Windows system settings.



The National Instruments Measurement and Automation Explorer also lists the instrument and shows the correct VISA resource string. This string needs to be used in the ARBToolbox in order to connect to the instrument.



In the example above the AFQ100A requires the VISA resource string 'USB0::0x0AAD::0x004B::100103::INSTR'. This resource may be used in the ARBToolbox for connecting to the instrument. Generally speaking the resource string for any AFQ100A type instrument is of the type USB::0xAAD::0x4B::<serial>::INSTR.

TCP/IP (VXI-11) Remote Control

The instrument can be connected to a LAN. By default the network settings are configured to receive the IP address via DHCP and all traffic is filtered by an internal firewall.

In order to connect to the instrument and remote control it by VISA the firewall must be deactivated. The following section describes all necessary steps for the SMU, SMJ and SMATE.

- Connect an English USB Keyboard and a USB mouse to the instrument.
- Move the mouse pointer to the lower boarder of the screen to bring up the Windows start menu.
- The topmost entry in the start menu has a name such as 'RSSMU200Annnnnn' where nnnnnn is the serial number. Make a note of this string. It will be the name that needs to be used to access the instrument.
- Click on the entry below (Control Panel) and then start the Windows Firewall configuration. For remote control the firewall must be entirely shut down (set to OFF). Close the control panel.
- Connect the LAN cable and make sure that your network is configured to use DHCP for automatic IP address assignment. Open a command prompt and try to ping your instrument. Use the network name you have noted above.

```
C:\> ping rssmu200a100123
```

If the connection is good there must be a response from the instrument.

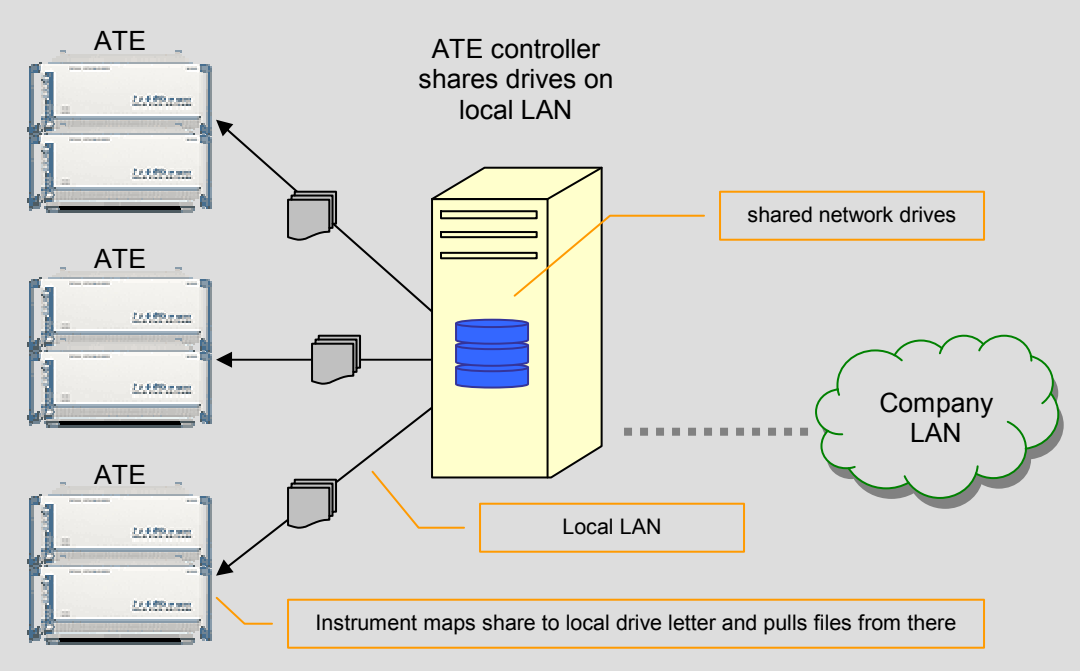
The following C code example shows how to use VISA to connect to the instrument. Please refer to the VISA documentation for further details.

```
viOpen (rmSession, "TCPIP::rssmu200a100123::INSTR", VI_NULL, VI_NULL,
        &instrSession);
```

Deactivating the firewall may be a potential risk since it exposes the instrument to the network and allows attacks and intrusion attempts. In a complex company network environment it is therefore

not recommended to shut down the firewall entirely. Please consult your the local network administration for instructions.

If security is an issue a possible solution could be to interface TCP/IP controlled systems through a PC that acts as ATE controller and firewall. Such a configuration could use local IP addresses within a local LAN and use DHCP with the external network.



Using Shared Drives

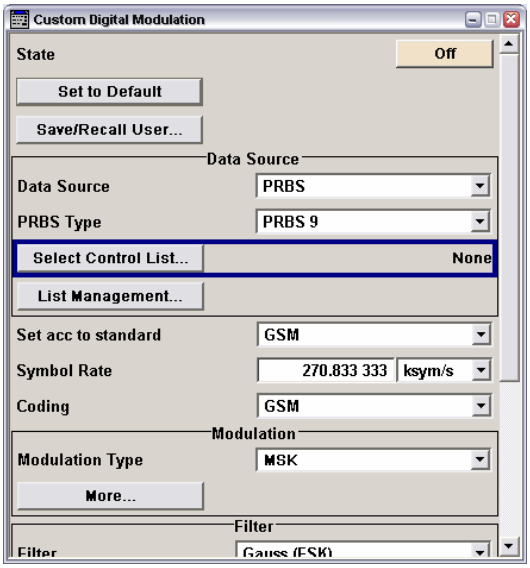
Once the instrument is connected to a network it is possible to share its D: drive. This allows to logon to the instrument and directly copy waveform or other files to the local file system. The default login and password is ‘instrument’. The operating manual describes the process of connecting to a TCP/IP network in great detail and also explains how the IP addresses could be changed.

This method is the fastest way to upload data files to the instrument because it directly copies data on the operating system level and thus avoids delays caused by the instrument firmware layer or additional drivers.

In an ATE setup like the one shown above it may also be desirable to share a drive that contains waveforms files and then map this share to a local drive letter on the instrument. The instrument ARB may then use this drive letter as source for its waveform files.

Manual Transfer with USB Sticks

Data can be copied to the signal generator using a USB stick. The simplest and recommended way of copying files is to use the built in file manager dialog. The mechanism is outlined below.

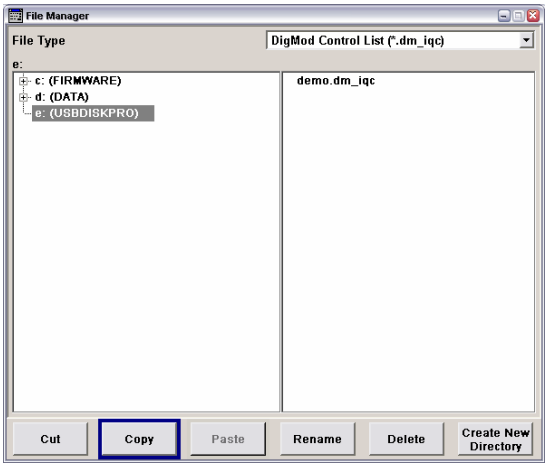


This example shows how to copy a waveform file from the USB stick to the instrument. It is not recommended to load the file directly from the stick since the file would not be available anymore if the stick was removed.

The starting point for the copy process is the 'Select Control List…' button after the stick was plugged in.

Pushing this button will bring up the file selection dialog. In the lower right corner is the button that starts the built in file manager.

Alternatively the 'FILE' button on the front panel can be pushed to bring up the save/recall dialog. The built in file manager can also be found in the lower right corner.



In the file manager dialog navigate to drive E:. Use the rotary knob, ENTER, ESC and the arrows for navigation.

Highlight the file that should be copied, then push the 'COPY' button. Now navigate to drive D:\ and push 'PASTE'.

After the copy process is done the USB stick can be removed and the file selected from the internal D:\ drive.

The internal file explorer does not allow to copy multiple files at a time. If many files should be copied a better approach would be to use the Microsoft Windows File Explorer instead. The process is described below:

- Connect an English USB keyboard and a USB mouse to the instrument.
- Plug the USB stick into one of the front panel USB ports. This will usually bring up the Windows task bar. If not, move the mouse cursor to the lower screen border to bring up the task bar.
- One of the icons of the menu bar is the Windows explorer. Open it and expand the item 'Instrument on RSSMU200A100022' where 100022 is an example for the instruments serial number.
- The USB stick shows up as drive E: and can be used as it can be with any general personal computer.

7 The ARB Toolbox

This application note comes with a tool box software that is called ARBToolbox. The program provides functions for the ARB mode, custom digital modulation as well as for the use of RF list mode. The following list briefly summarize some of the ARBToolbox features.

- Connect to instrument via VISA resource (GPIB, TCPIP)
- Log window shows all communication with instrument
- Run SCPI commands from a text file (batch mode)
- Create simple wave forms such as AM, FM and FM chirp and save wave file to local disk
- Upload wave file to instrument and activate playback if needed
- Import IQ data from text file or binary files and generate wave file from data
- Create data and control list from data for use as a simple example
- Upload data and control list to the instrument and activate playback if required
- Import list data from simple text file and create data and control list
- Create RF list from table data or from imported data, activate if required
- Browse hard drive on instrument and allow to copy files to local PC

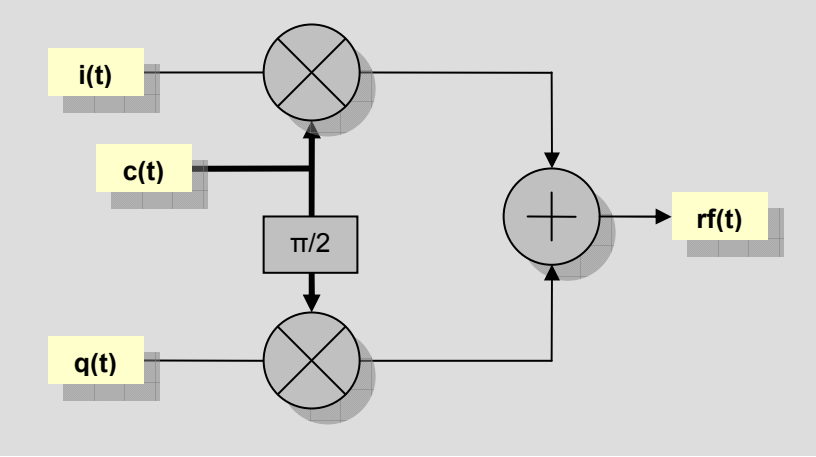
The foregoing paragraph does first discuss some basic theory about IQ modulation. This reading is required to understand the signal creation functionality of the ARBToolbox. However, if the user is already familiar with the concepts of IQ modulation the theory section may be skipped.

Example: How to generate an FM chirp signal

The I and Q signals in the example are simple sine waves and can be described by the following formulas. C(t) describes the RF carrier signal.

$$i(t) = A \cdot \sin(\omega t + \phi) + \beta$$
$$q(t) = \sin(\omega t - 90^\circ)$$
$$c(t) = \cos(\omega_c t)$$
$$with\ \omega = 2\pi f$$

The IQ modulator multiplies i(t) with the carrier c(t) and multiplies q(t) with the phase shifted carrier. The sum signal rf(t) is the sum of both multiplication results. In our example the variable A is the amplitude ratio between i(t) and q(t). Φ defines the phase offset and β represents a DC offset.



Block chart of IQ modulator

The resulting waveform $rf_{mod}(t)$ can be expressed as follows:

$$rf_{\mathrm{mod}}(t) = A \cdot \sin(\omega t + \phi)\cos(\omega_c t) + \beta \cdot \cos(\omega_c t) - \cos(\omega t)\sin(\omega_c t)$$

This equation shows that the carrier signal $\cos(\omega_c t)$ only exists if a DC offset β is present. For the following evaluation the DC offset is assumed to be zero. With the two sin-cos relationships below the equation for $rf_{mod}(t)$ can be simplified.

$$\sin(\alpha)\cos(\beta) = \frac{1}{2}\sin(\alpha-\beta) + \frac{1}{2}\sin(\alpha+\beta) \quad ,$$
$$\sin(\alpha+\beta) = \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta)$$

$$rf_{\mathrm{mod}}(t) = \frac{A}{2}\sin[(\omega-\omega_c)t] + \frac{A}{2}\sin[\phi + (\omega+\omega_c)t] - \frac{1}{2}\sin[(\omega_c-\omega)t] - \frac{1}{2}\sin[(\omega_c+\omega)t]$$

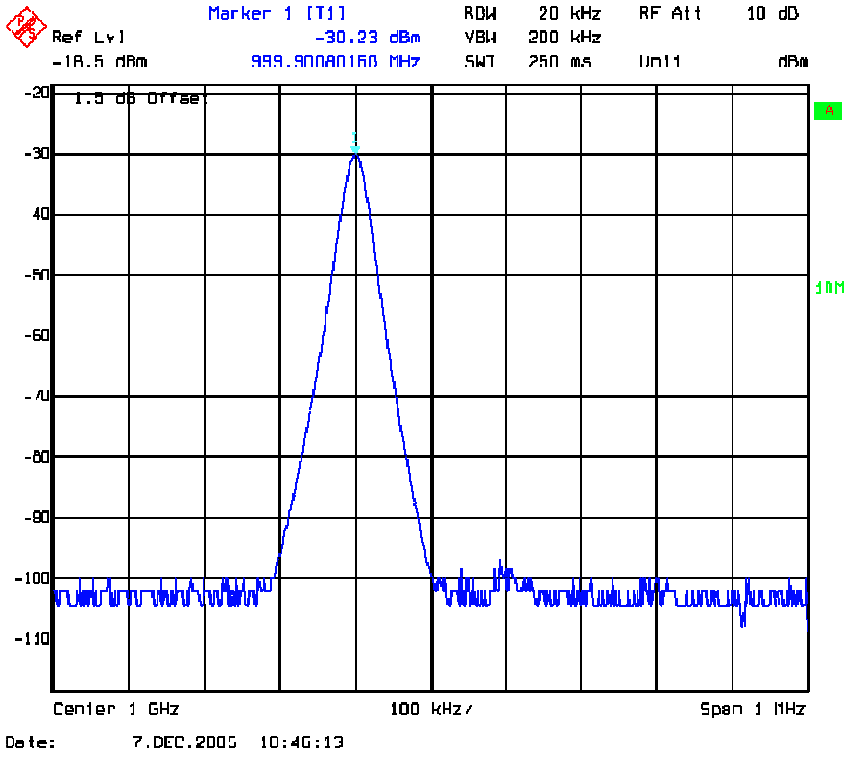
Rearranging the terms in this equation unveils the relationships for the upper side band USB ($\omega_c+\omega$) and lower side band LSB ($\omega-\omega_c$).

$$USB: \quad rf_{\mathrm{mod}}(t) = \frac{A}{2}\sin\left[(\omega+\omega_c)t\right]\cdot\cos(\phi) + \frac{A}{2}\cos\left[(\omega+\omega_c)t\right]\cdot\sin(\phi) - \frac{1}{2}\sin\left[(\omega+\omega_c)t\right]$$

$$LSB: \quad rf_{\mathrm{mod}}(t) = \frac{A}{2}\sin\left[(\omega-\omega_c)t\right]\cdot\cos(\phi) + \frac{A}{2}\cos\left[(\omega-\omega_c)t\right]\cdot\sin(\phi) + \frac{1}{2}\sin\left[(\omega-\omega_c)t\right]$$

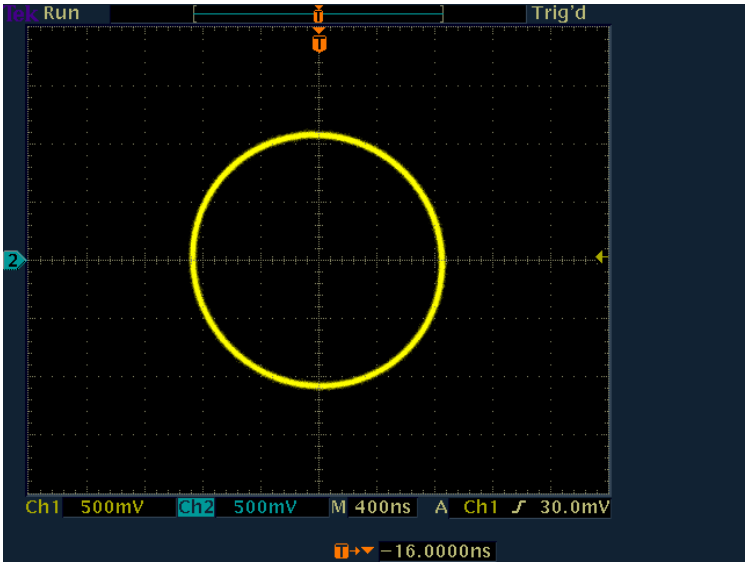
The formulas indicate that the resulting signal contains weighted sine waves of the two frequencies ($\omega-\omega_c$) and ($\omega+\omega_c$). The ratio depends on A and cos(Φ).

For A=1 and a phase shift of 90 degrees the USB is zero whereas the LSB resolves to weighted sin(($\omega-\omega_c$)t). The resulting spectrum shows a peak at -100 kHz (the modulation frequency) away from the carrier at -30 dBm RF power.



Screen shot from spectrum analyzer, A=1, Phase Shift 90 degrees

The following screen shot shows the IQ vector diagram with both amplitudes set to one and the phase shift at 90 degrees.



Screen shot in IQ plane

Let's look at the LSB equation a bit closer. The simplified equation for A=1 and a phase shift of 90 degrees resolves to:

$$LSB: \; rf_{\mathrm{mod}}(t) = \frac{1}{2}\cos[(\omega-\omega_c)t]+\frac{1}{2}\sin[(\omega-\omega_c)t] = \cos(\frac{\pi}{4})\cdot\sin[(\omega_c-\omega)t]$$

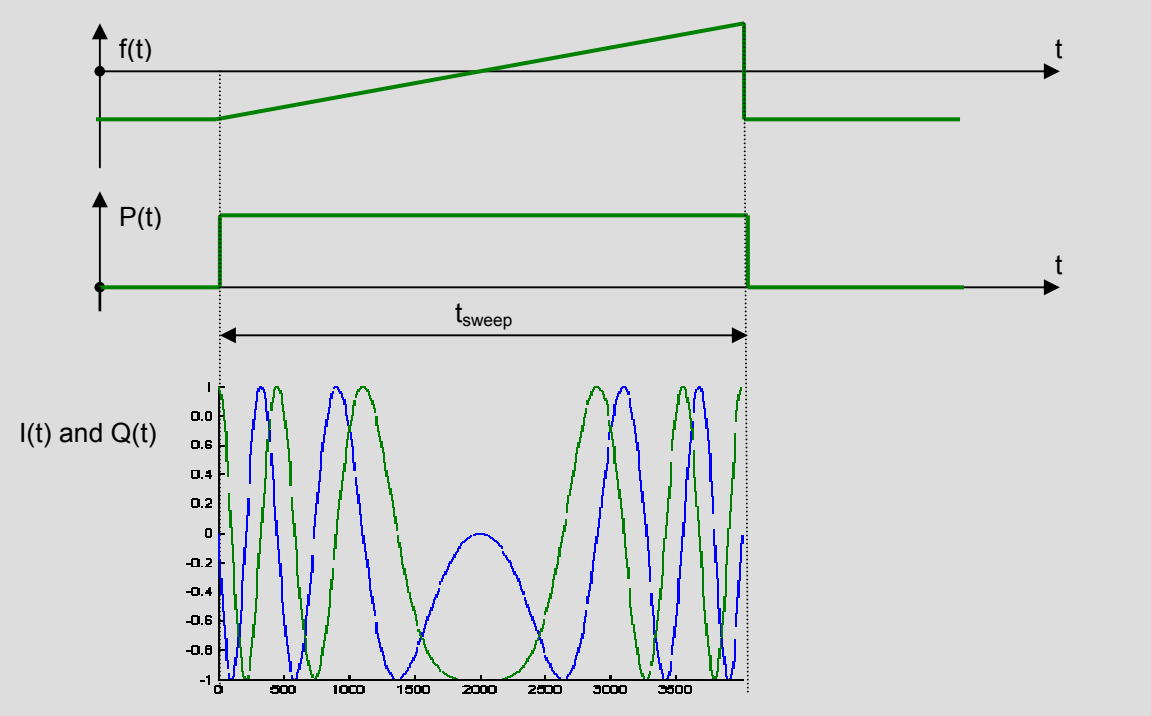
This demonstrates that the i(t) and q(t) signals from above create a simple LSB that consists of one single, weighted frequency. If the frequency ω of the I and Q signals vary with time it is possible to create a frequency modulated output signal.

The equation below can be used for the generation of a frequency modulated chirp for a given time 0<=t<=t_{sweep} with f being the frequency shift.

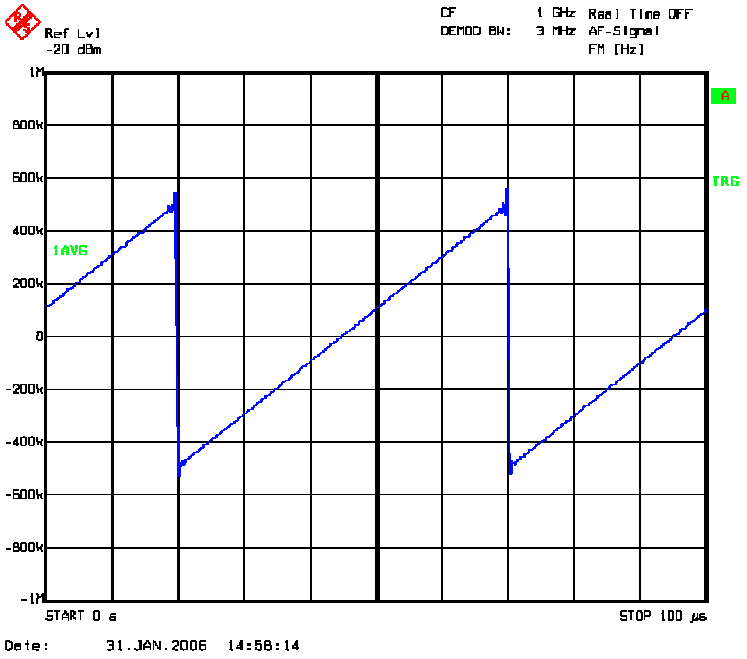
$$i(t) = \sin\left[2\cdot\pi\cdot f(-1+\frac{t}{t_{sweep}})\cdot t\right] \;,$$

$$q(t) = \sin\left[2\cdot\pi\cdot f(-1+\frac{t}{t_{sweep}})\cdot t + \frac{\pi}{2}\right]$$

The ARB Toolbox program can generate a chirp waveform of 50 µs duration with a frequency shift of +/- the specified frequency.



I and Q relationships during FM chirp



Example FM chirp

Installing the ARB Toolbox

The toolbox comes as a ready to use installer. However, there are certain prerequisites for running the program.

Most importantly VISA needs to be installed on the PC. It can be obtained on the National Instruments web site and is also bundled with their GPIB hardware. Please see the web site for more details and the license terms.

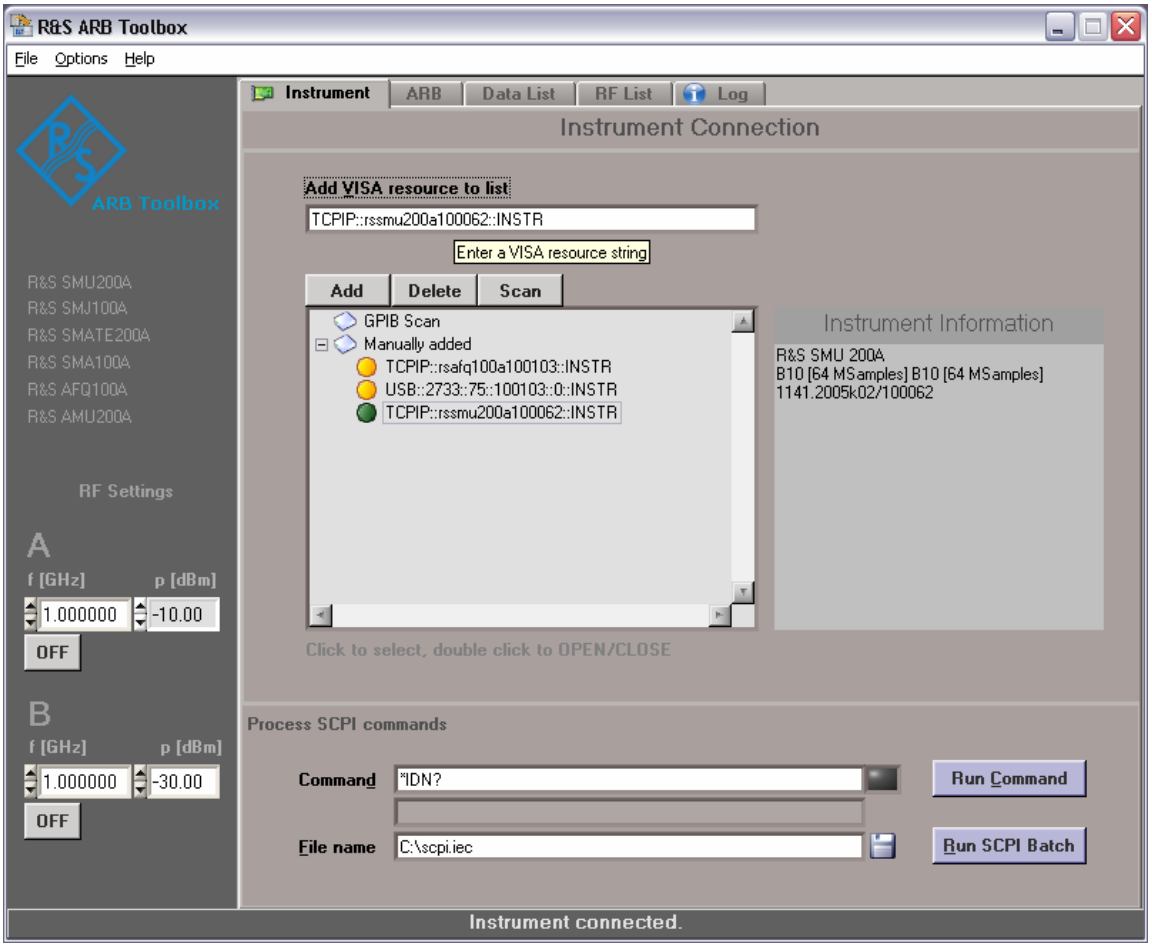
Once these requirements are met the toolbox can be installed by simply starting the installer. Follow the instructions and after completion you will find the ARB Toolbox program as a new entry in your Windows start menu.

Besides the installer some text files that demonstrate the import functionality are bundled in the ZIP package.

File	Description
ARBToolbox_x.y.z.exe	Installer with serial number x.y.z
demo.iec	Script file with SCPI commands
dlist-import.txt	Text file for use with data and control list import
rflist-import.txt	Text file for use with RF list import
rs_demo.m	Matlab demo file for use of the Matlab toolbox
wave-import.txt	Text file for use with waveform import

Using the ARBToolbox

The program starts with the instrument connection panel. This panel is used to scan the local bus as well as for setting up the instrument link.



Screen shot of instrument control panel

Pushing the 'scan' button starts a local bus scan and shows all discovered devices in the list box. Supported instruments are marked green whereas unsupported devices show up in red.

If connections other than the local bus shall be used it is possible to add the VISA resource string to the list box. All manually added items are marked blue. The example shows a TCP/IP connection that was added manually.

Clicking one of the blue or green lines establishes the connection, evaluates the instrument capabilities and displays some device information in the information area. Subsequently, the folder icon changes to an opened one to indicate that the connection was successfully established.

Currently the ARB toolbox program supports the instruments SMU200A, SMJ100A and SMATE200A.

All open connections are automatically closed when the user quits the program.

Running SCPI batch scripts

It is possible to process files that contain SCPI commands in batch mode. This feature is very useful for the development of own applications and to experiment with remote control commands.

The syntax of the SCPI batch files is extremely simple. All lines that do not start with '%' or '/' are interpreted as command and the content is send to the instrument. Commands can optionally be terminated with *OPC?. Queries will also work and the result appears in the log window.

In addition, an extra command called @wait is possible with a waiting time specified in milliseconds.

```
%
% set some RF parameters and activate output signal

:SOUR:BB:DM:PRES
:SOUR:FREQ 1.2 GHz; :POW -30.0 dBm; *OPC?
:OUTP:STAT ON; *OPC?

:STAT:OPER?
@wait=2000
```

If an error occurs processing is stopped and a message is shown in the status line. More detailed information can be obtained from the activity log window.

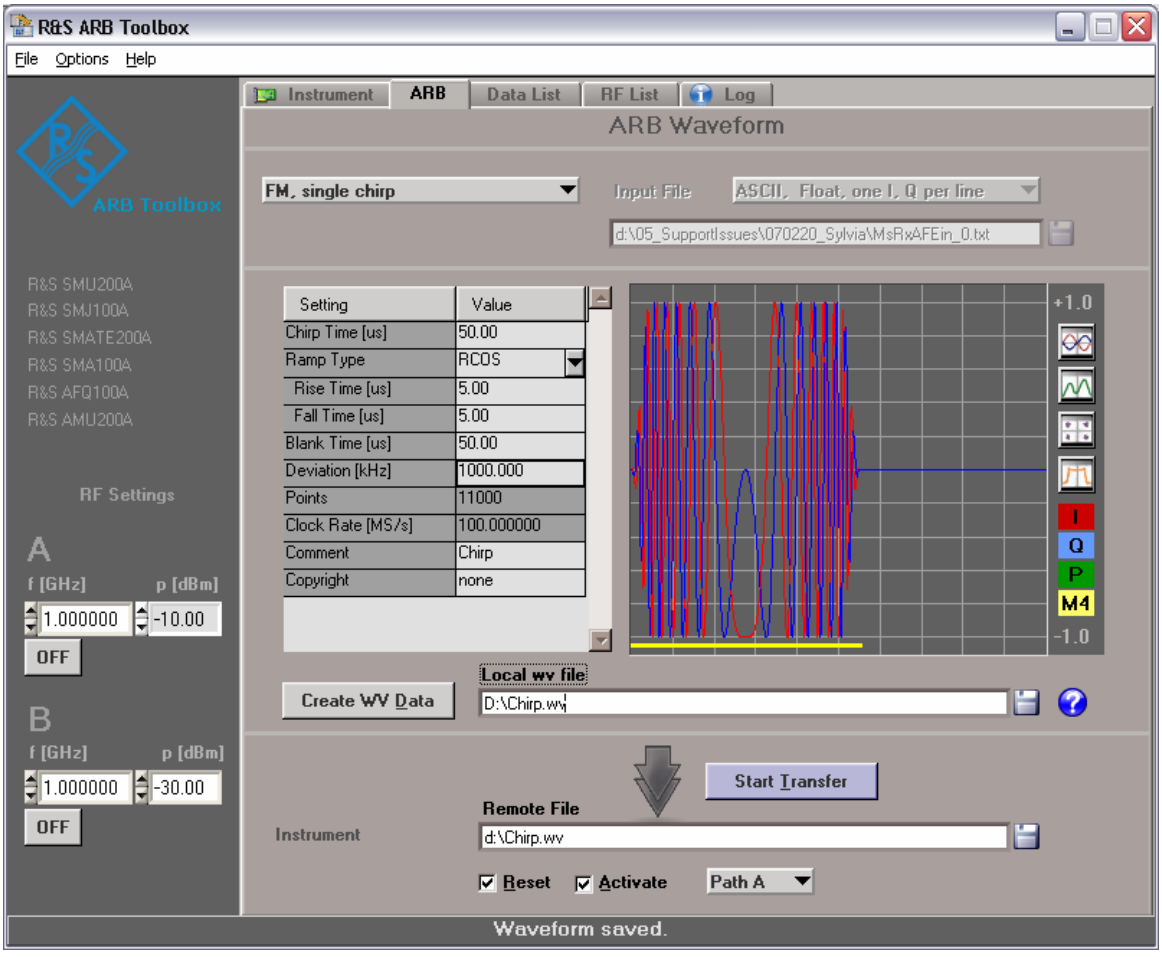
The information on this panel is for command reference as well as for error tracking. The GPIB commands can be copied (Control-C) and used in own applications.

Working with the ARB Panel

The ARB panel provides useful functionality for the following tasks:

- Generate sample data, either simple sine waves or an FM chirp
- Import custom IQ data from text file or from binary data
- Generate wave file on local hard drive
- Upload local wave file to instrument and activate playback
- Copy wave file from the instrument to a local hard drive

The screen shot below shows the ARB panel when used for generating the FM chirp. The very general workflow in all program modules is from left to right and from top to bottom.



Screen shot of ARB panel

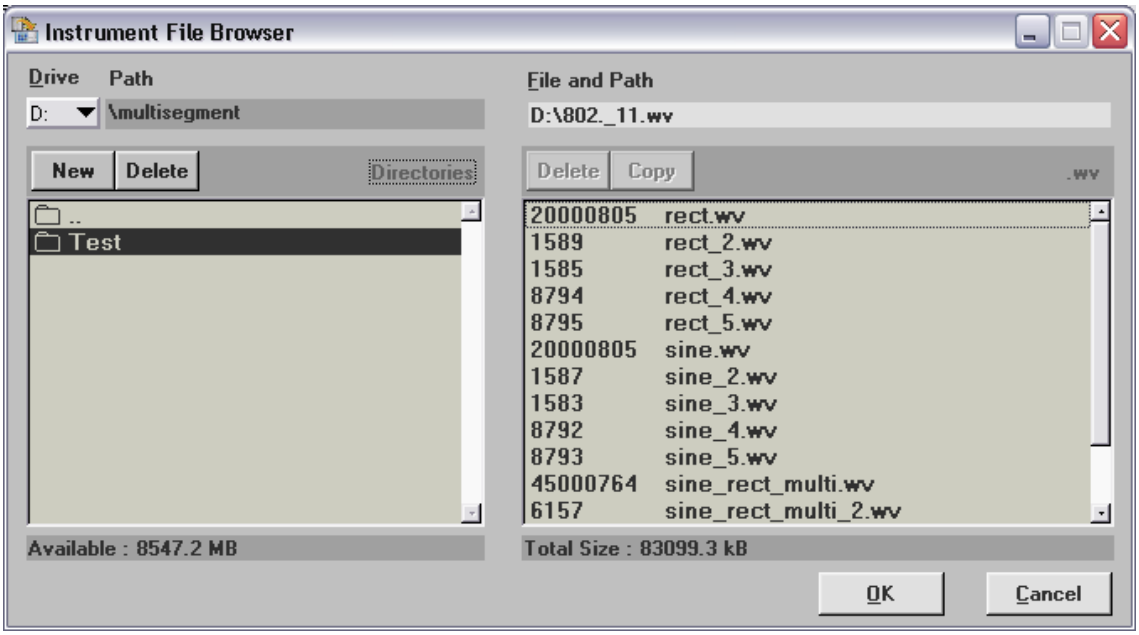
First, one need to select whether to use the dialog in data creation or import mode. This can be achieved with the two top most radio buttons. In data creation mode almost all entry fields of the table on the left side are available. A double click on a table cell switches this cell to edit mode.

A click on the 'Create' button draws the I and Q waveforms and creates the local wave file.

The waveform file of the example sets marker 1 to high for one data byte at the beginning of a period. This mechanism can be used if more complex waveforms are generated and other instruments need to be triggered.

In import mode most fields of the table on the left side are grayed out, even though the principle work flow remains the same. Clicking 'Import' start the import process and generates the local wave form file.

The next step would be to upload the waveform to the instrument and activate its playback. The upload always uses the local waveform file source and the instrument file as destination. The small floppy disk button right of the instrument file name starts a browser for the remote file system.



Browser for remote file system

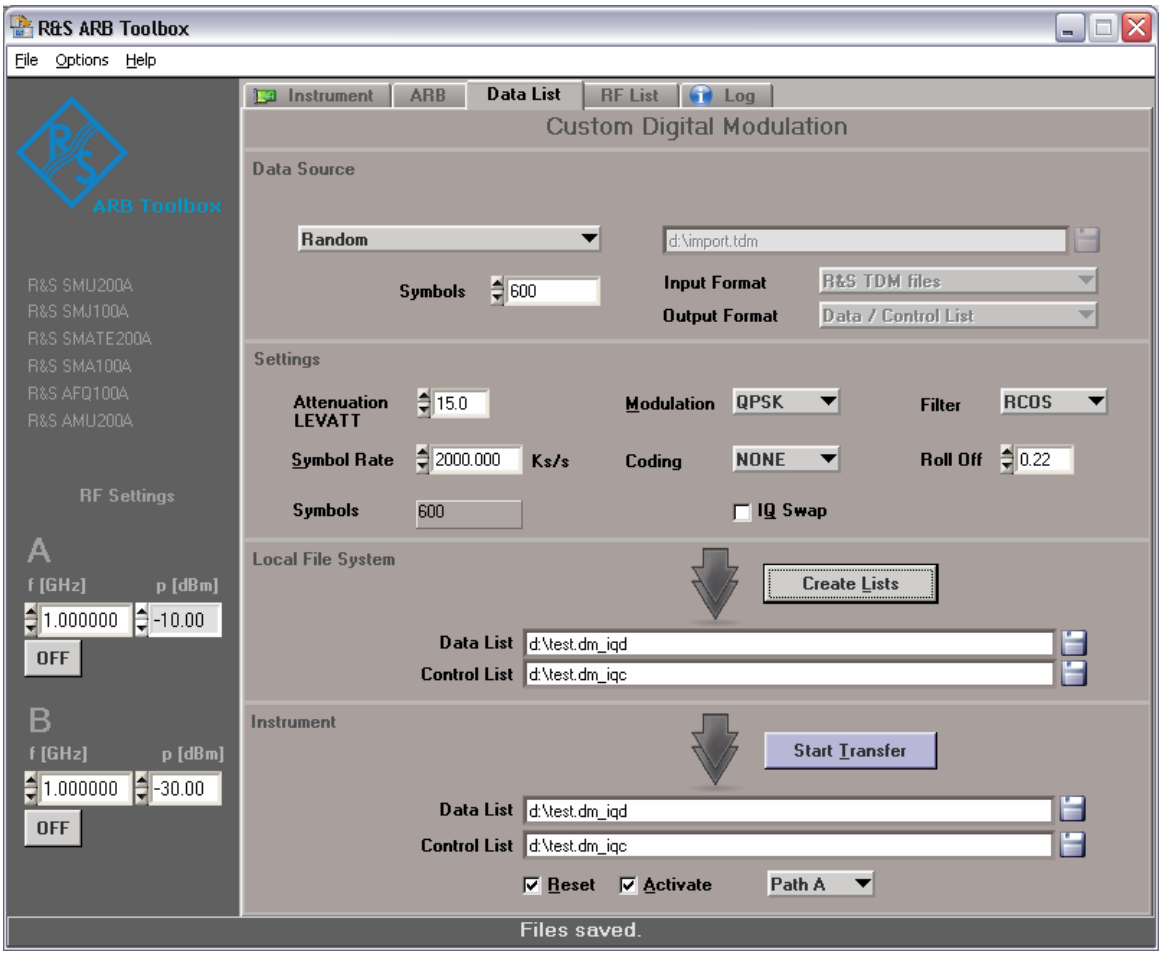
The dialog is useful for many purposes. First, it can simply be used to specify the destination directory and file. A file name can be entered manually and accepted by clicking 'OK'. It is also possible to navigate to sub directories or create them as required. If existing files are selected their content will be overwritten.

The button 'Copy to local PC' brings up a second dialog to specify the destination on the local file system. This functionality can be used to backup existing files from an instrument.

The upload can be started via the 'Upload' button on the ARB panel. The RF settings for frequency and level are applied if activation of the uploaded data is chosen.

Working with Custom Digital Modulation

The following screen shot shows the custom digital modulation panel in data creation mode. The principal workflow is very similar to the ARB panel.



Screen shot of data list panel

In data creation mode a data pattern and a number of symbols must be selected as input for the data list. The actual content is compiled with the knowledge of the modulation settings from the box below. These settings, however, do not directly affect the list content but the modulation scheme defines the number of bits that are required per symbol.

Hitting the 'Create' button writes the data list as well as an associated control list to the local hard drive.

Alternatively data can be imported from a text file as input for the creation of the data and control list. The text format is relatively simple and is described below. In import mode all settings such as attenuation, symbol rate, modulation and filter settings from the panel serve as defaults and can be overwritten by data entries from the import file.

Clicking 'Upload' starts the transfer of both files to the instrument. The small '<' buttons right next to the instrument file name entry lines start the remote file system browser that was discussed before.

Importing data from text files is very simple. The only requirement regarding the file format is that all data is prevalent as sequences of zeros and ones. The number of bits per line should be a multiple of the number of bits defined by the modulation scheme.

Lines that start with '%' or '/' are treated as comments and ignored. Lines starting with the exclamation mark are evaluated and may be used to overwrite default settings from the dialog panel.

At the end of each data line a keyword may be appended to activate marker signals or setup the burst structure. This is the major reason why the number of bits per line must be a multiple of the number of bits per symbol used by the modulation scheme.

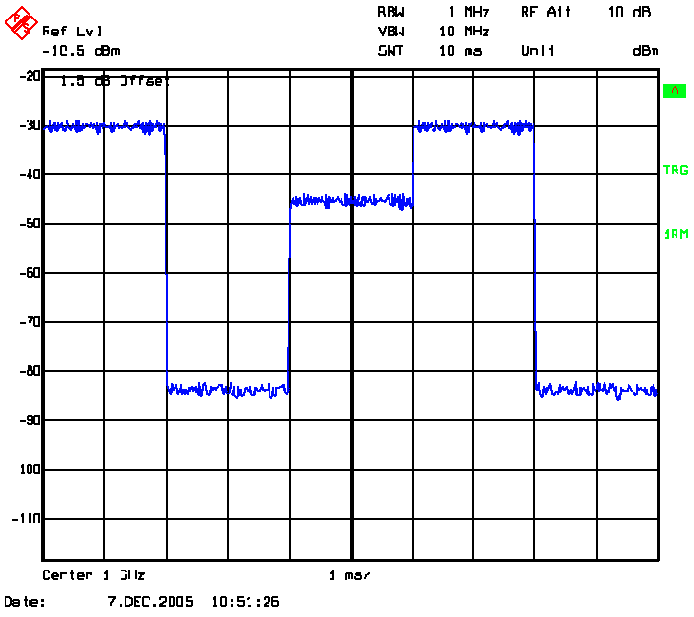
The keywords do not necessarily need to be separated by colons but it increases the files readability.

```
% demo for data import
!bitspersymbol = 2
!mod           = QPSK
!comment       = demo list
!copyright     = none
!att           = 15.0
!clockrate     = 100000
!filter        = RCOS
!alpha         = 0.22


00                          BURST,MARKER1
110011   10011100           BURST
```

The screen shot below shows a waveform generated by importing the demo text data.

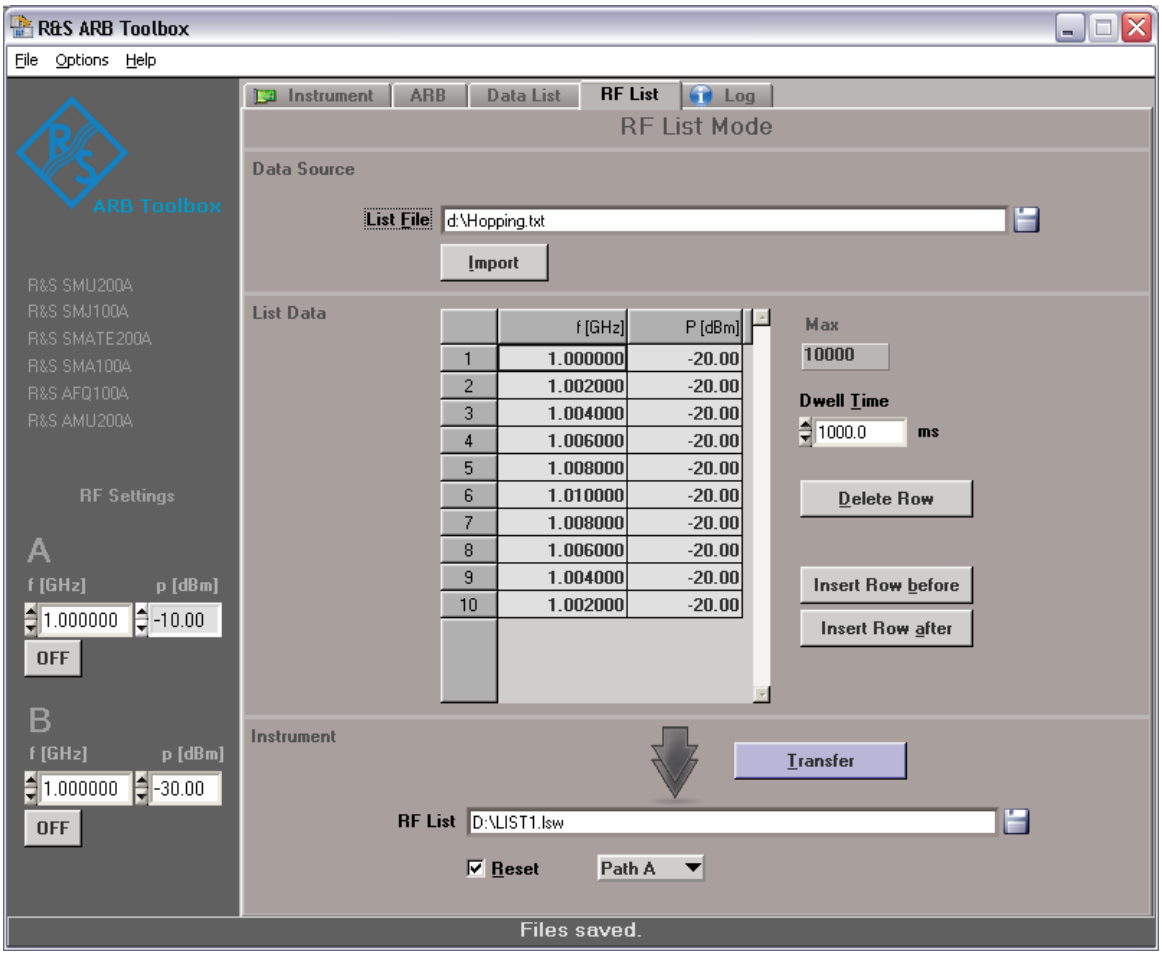
The control list generates a marker signal on the very first symbol which can be used to trigger a spectrum analyzer. This could be done by connecting "MARKER 1" of the front panel to the external trigger input of the spectrum analyzer. If base band B is used MARKER 1b is located on the rear panel of the instrument.



Screen shot of RF output

Working with the RF List Mode Panel

The RF list panel allows to specify 10 RF frequency and level pairs. The following screen shot shows the dialog in its default configuration.



Screen shot of frequency list panel

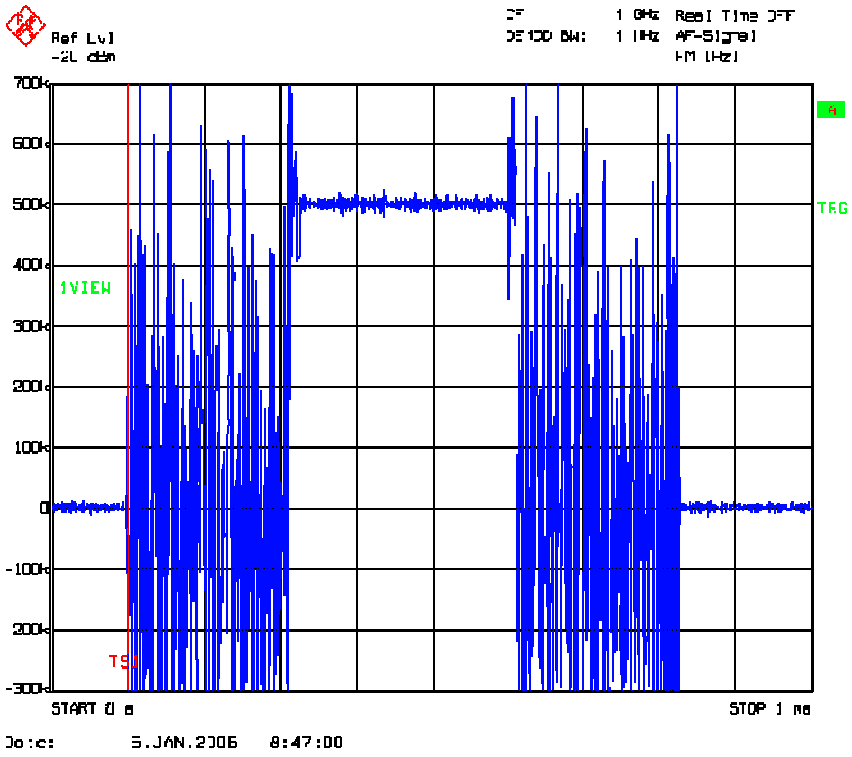
The table contains 10 frequency / level pairs which are played back at the rate of the given dwell time.

In contrast to the other operating modes the frequency list is not generated offline and then uploaded to the instrument. One of the reason is that RF lists need to be generated on the instrument as they contain device specific information in order to increase switching speed.

The activity log shows all GPIB commands that were used and could serve as a guideline for own implementations.

The RF default settings do not have any effect in this dialog.

The screen shot below shows the FM demodulated output signal of the SMU while hopping from 1.0 GHz to 1.0005 GHz. The trigger (red line at + 0.1 ms) is set to the rising edge of the blank flag which was routed to user output one. In the example the dwell time was set to 0.5 ms and includes the blanking period at the very beginning.



Screen shot of RF frequency hop in FM mode

The guaranteed time for a frequency hop is below 450 µs for SMU 200A. During frequency or level change the signal is blanked to avoid over shots that may be a potential risk to connected circuitries. The BLANK signal is activated during blanking periods and can be routed to the user outputs of the instrument.

8 Code Examples

Mat Lab Example for Use with Matlab Toolbox

```
% function [Status,InstrObj] = rs_demo(varargin)
% ************************************************************************
%
% Copyright: (c) 2005 Rohde & Schwarz GmbH & CO KG.    All rights reserved.
%                Muehldorfstr. 15
%                D-81671 Munich
%
% ************************************************************************
% This function is used to demonstrate the Matlab Toolbox for the
% purpose of generating and uploading a waveform to the instruments
% SMU, SMJ, SMATE.
%%
% @param            none
% @return Status    none
%************************************************************************

function [Status,InstrObj] = rs_demo(varargin)

    % -----------------------------------
    % Connect to GPIB Instrument on
    % address 28
    % -----------------------------------
    [status, object] = rs_connect('GPIB', 0, 28);
    if (status<1)
      return;
    end

    % -----------------------------------
    % Send ID query to verify Instrument,
    % then reset it
    % -----------------------------------
    [status, answer] = rs_send_query(object, '*IDN?');
    if (status<1)
      return;
    end
    answer

    disp ('Resetting...');
    [status] = rs_send_command(object, '*RST');
    if (status<1)
      return;
    end

    % -----------------------------------
    % Apply RF settings
    % -----------------------------------
    [status] = rs_send_command(object, 'FREQ 1.0 GHz;*OPC');
    if (status<1)
      return;
    end
    [status] = rs_send_command(object, 'POW -30.0 dBm;*OPC');
    if (status<1)
      return;
    end
```

```
    % ----------------------------------
    % Set Modulation parameters
    % O    = Offset I
    % A    = Amplitude I
    % Phi  = Phase Shift I
    % Freq = Frequency
    % OS   = Oversampling Ratio
    % ----------------------------------
    O    = 0.0;
    A    = 1.0;
    Phi  = 0.0;
    Freq = 100e3;
    OS   = 512;

    % ----------------------------------
    % Check if OS*Freq exceeds the maximum
    % clock rate of the instrument
    % ----------------------------------
    if (OS*Freq > 100e6)
      error ('Error: Max. clock rate exceeded !');
      return;
    end

    % ----------------------------------
    % Generate time ticks for IQ waveform
    % and fill waveform structure for Tool Box
    % ----------------------------------
    k = [0:1:OS-1];
    Time = k * (1.0/Freq) / OS;

    IQInfo.I_data    = A * sin(2.0*pi*Freq*Time+   Phi) + O;
    IQInfo.Q_data    =     sin(2.0*pi*Freq*Time-pi/2.0);
    IQInfo.comment   = 'ARB Example';
    IQInfo.clock     = OS * Freq;
    IQInfo.path      = 'D:\';
    IQInfo.filename  = 'demo.wv';
    IQInfo.markerlist.one = [[0 1];[1 0]];

    % ----------------------------------
    % Plot in IQ plane
    % ----------------------------------
    subplot(2,1,1);
    hold all
    grid on
    plot (IQInfo.I_data);
    plot (IQInfo.Q_data);
    subplot(2,1,2);
    plot (IQInfo.I_data, IQInfo.Q_data);

    % ----------------------------------
    % Send data to instrument and activate
    % ----------------------------------
    [status] = rs_generate_wave(object, IQInfo, 1, 1);
    if (status<1)
      return;
    end

    disp ('Waveform created and uploaded.');

return;
```

Lab Windows/CVI C Code examples

Function 1: Opening a VISA Session

```
#include <userint.h>
#include <formatio.h>
#include <utility.h>
#include <ansi_c.h>
#include <visa.h>
#include <math.h>

#define   PI   3.141592654

#define   INVALID_SESSION  -1
ViSession rmSession    = INVALID_SESSION;
ViSession instrSession = INVALID_SESSION;

/**
 * Open VISA connection.
 * This function opens a visa session.
 * \param sRscName = VISA resource string
 *                   GPIB::<n>::INSTR -> GPIB device on primary address n
 *                   TCPIP::<x>::gpib<b>,<d>::INSTR -> TCP/IP converter, e.g. AD007
 *                   TCPIP::<type><serial>::INSTR   -> TCPIP connection, e.g.
 *                    rssmu200a100052, firewall needs to be deactivated
 */
ViBoolean arbDemoLibOpen (const char* sRscName)
{
   int iStatus = 0;

   // check if device is already open
   if (instrSession != INVALID_SESSION)
      return VI_FALSE;
   // open default resource manager and instrument session
   if (viOpenDefaultRM (&rmSession) < 0)
      return VI_FALSE;
   if ((iStatus=viOpen (rmSession, (char*)sRscName, VI_NULL, VI_NULL,
         &instrSession)) != VI_SUCCESS) {
      viClose (rmSession);
      instrSession = rmSession = INVALID_SESSION;
      return VI_FALSE;
   }
   // configure VISA formatted I/O
   do {
      // set timeout to 5000 ms, should be sufficient for most applications
      if ((iStatus = viSetAttribute (instrSession, VI_ATTR_TMO_VALUE, 5000)) < 0)
         break;
      // set input and output buffer sizes to 2 KByte
      if ((iStatus = viSetBuf (instrSession, VI_WRITE_BUF, 2048)) < 0)
         break;
      if ((iStatus = viSetBuf (instrSession, VI_READ_BUF, 2048)) < 0)
         break;
      if ((iStatus = viSetAttribute (instrSession, VI_ATTR_WR_BUF_OPER_MODE,
            VI_FLUSH_ON_ACCESS)) < 0)
         break;
      if ((iStatus = viSetAttribute (instrSession, VI_ATTR_RD_BUF_OPER_MODE,
            VI_FLUSH_ON_ACCESS)) < 0)
         break;
   } while (VI_FALSE);
   // clean up
   if (iStatus < 0)  {
      viClose (instrSession);
      viClose (rmSession);
      instrSession = rmSession = INVALID_SESSION;
      return VI_FALSE;
   }

   return VI_TRUE;
}
```

Function 2: Closing a VISA Session

```
/**
 * Close VISA connection.
 * This function closes the VISA session.
 */
ViBoolean arbDemoLibClose (void)
{
  // check for valid session handle
  if (instrSession == INVALID_SESSION)
    return VI_FALSE;
   // set instrument back to local mode
   viGpibControlREN(instrSession, VI_GPIB_REN_DEASSERT_GTL);

   if (instrSession != INVALID_SESSION)
      viClose (instrSession);
   if (rmSession != INVALID_SESSION)
      viClose (rmSession);

   // invalidate internal variables
   instrSession = INVALID_SESSION;
   rmSession = INVALID_SESSION;

   return VI_TRUE;
}
```

Function 3: Sending a SCPI command to the instrument (write)

```
/**
 * Send command.
 * This function sends a command to the instrument. If the command is
 * terminated with *OPC? A read is automatically executed.
 * \param sOutBuf = output text string
 */
ViBoolean arbDemoLibWrite (const char *sOutBuf)
{
   int      iStatus = 0;
   ViUInt32 iRead   = 0;
   char     cRecBuf[33];

   // check for valid session handle
   if (instrSession == INVALID_SESSION)
      return VI_FALSE;
   // send data as text string
   if (strlen(sOutBuf)!=0) {
      if ((iStatus = viPrintf (instrSession, (char*)sOutBuf)) < 0)
         return VI_FALSE;
   }
   // terminated with *OPC?
   if (strstr(sOutBuf, "*OPC?")!=NULL) {
      memset (cRecBuf, 0, sizeof(cRecBuf));
      // receive OPC status information
      if ((iStatus = viRead (instrSession, (ViPBuf)cRecBuf,
            sizeof(cRecBuf)-1, &iRead)) < 0)
         return VI_FALSE;
      cRecBuf[iRead]=0;
      // check result
      if (cRecBuf[0]!='1')
         return VI_FALSE;
   }

   return VI_TRUE;
}
```

Function 4: Sending binary data block wise to the instrument

```
/**
 * Write binary data
 * This function sends binary data to the instrument.
 * \param cOutBuf = pointer to output buffer
 * \param iBufLen = length of output buffer
 */
ViBoolean arbDemoLibWriteBin(const char *cOutBuf, int iBufLen)
{
   int                iBytesRemain, iBytesToSend, iStatus = VI_TRUE;
   ViUInt32           iSentBytes;
   ViBoolean          bEOIStatus;
   unsigned int       iRead;

   // check for valid session handle
   if (instrSession == INVALID_SESSION)
      return VI_FALSE;
   // send data as binary
   if (iBufLen!=0 && cOutBuf!=NULL) {
      // use EOI line instead of character
      viGetAttribute(instrSession, VI_ATTR_SEND_END_EN, &bEOIStatus);
      viSetAttribute(instrSession, VI_ATTR_SEND_END_EN, VI_FALSE);
      iBytesRemain = iBufLen;
      while (iBytesRemain>0) {
         iBytesToSend = iBytesRemain;
         if (iBytesToSend>2048)
            iBytesToSend = 2048;
         // last transfer
         if (iBytesRemain<=iBytesToSend)
            viSetAttribute(instrSession, VI_ATTR_SEND_END_EN, VI_TRUE);
         // send out one block
         iStatus = viWrite (instrSession, (ViBuf)cOutBuf, iBytesToSend, &iSentBytes);
         if (iStatus < 0 || iBytesToSend!=iSentBytes)
            return VI_FALSE;
         iBytesRemain -= iBytesToSend;
         cOutBuf += iSentBytes;
      }
      // set EOI back to its old state
      viSetAttribute(instrSession, VI_ATTR_SEND_END_EN, bEOIStatus);
   }

   return VI_TRUE;
}
```

The waveform data structure

```
/**
 * SMU wave file parameters
 * This structure is used with arbDemoLib_SMU_GenerateWvFile
 */
typedef struct
{
   unsigned int  iClockRate;  // clock rate in Hz
   unsigned int  iPoints;     // number of points
   double        dRmsOfs;     // RMS offset in dB
   double        dPeakOfs;    // Peak offset in dB
   char          *szComment;  // comment
   double        *dI;         // buffer for I values
   double        *dQ;         // buffer for Q values
   unsigned char *ucFlags;    // buffer for control flags (NULL if unused)
} T_WV_DATA;
```

Function 5: Generating a waveform file from existing I/Q data

```
/**
 * Create SMU waveform file from data.
 * This function creates a waveform file on the local hard drive.
 * \param tWvData = waveform data structure
 * \param szFile = file name of output file
 */
ViBoolean arbDemoLib_GenerateWvFile(T_WV_DATA *tWvData, const char *szFile)
{
   ViInt32     i,j;                    // counter variable
   ViInt16     _uiI, _uiQ;             // IQ values as integer
   FILE        *fp;                    // file pointer
   ViInt32     _iYear, _iMonth, _iDay; // system date
   ViInt32     _iH, _iMin, _iSec;      // system time
   ViChar      szTmp[256];
   ViInt16     iOld, iState;

   if (strlen(szFile)==0 || tWvData->iPoints == 0)
      return VI_FALSE;
   // create waveform file
   // !!! must be in binary format !!!
   fp = fopen (szFile, "wb");
   if (fp==NULL)
      return VI_FALSE;
   // write header and comment
   fprintf (fp, "{TYPE: SMU-WV,0}");
   fprintf (fp, "{COMMENT: %s}", tWvData->szComment);
   // write time stamp
   GetSystemDate (&_iMonth, &_iDay, &_iYear);
   GetSystemTime (&_iH, &_iMin, &_iSec);
   fprintf (fp, "{DATE: %04d-%02d-%02d;%02d:%02d:%02d}"
             ,_iYear, _iMonth, _iDay, _iH, _iMin, _iSec);
   fprintf (fp, "{CLOCK: %d}"          , tWvData->iClockRate);
   fprintf (fp, "{LEVEL OFFS: %f,%f}"  , tWvData->dRmsOfs, tWvData->dPeakOfs);
   fprintf (fp, "{SAMPLES: %d}"        , tWvData->iPoints);
   // write marker flags
   if (tWvData->ucFlags != NULL) {
      // control list length
      fprintf (fp, "{CONTROL LENGTH: %d}", tWvData->iPoints);
      // write control block for markers 1...4, etc.
      for (i=0; i<4; i++) {
         fprintf (fp, "{MARKER LIST %d: ", i+1);
         iOld = -1;
         // all data points
         for (j=0; j<tWvData->iPoints; j++) {
            iState = tWvData->ucFlags[j] & (unsigned char)pow(2,i);
            if (iOld != iState) {
               if (iOld!=-1) fprintf (fp,";");
               fprintf (fp, "%d:%d", j, iState==0?0:1);
               iOld = iState;
            }
         }
         // terminate data section
         fprintf (fp, "}");
      }
   }
   // write waveform block
   fprintf (fp, "{WAVEFORM-%d: #", tWvData->iPoints*4+1) ; // each pair is 4 bytes
   for (i=0; i<tWvData->iPoints; i++) {
      // _dI, _dQ values are +/-1.0
      _uiI = (ViInt16)(tWvData->dI[i]*32767.0+0.5);
      _uiQ = (ViInt16)(tWvData->dQ[i]*32767.0+0.5);
      fwrite (&_uiI,1,2,fp); fwrite (&_uiQ,1,2,fp);
   }
   fprintf (fp, "}");
   // close file
   fclose (fp);

   return VI_TRUE;
}
```

Function 6: Uploading the waveform file to the instrument

```
/**
 * Upload a waveform file.
 * This function uploads a waveform file from the local hard drive
 * to the instrument. After the upload the file can be activated.
 * \param szFile = input file from local hard drive
 * \param szInstrFile = file on instrument
 * \param iBBPath = base band path 1 or 2
 * \param bActivate = activate after upload
 */
ViBoolean arbDemoLib_UploadWvFile(const char *szFile, const char *szInstrFile,
                                  int iBBPath, ViBoolean bActivate)
{
   FILE            *fp;
   int             iFileSize;       // file size (input file)
   char            *pcBuf,*pcBuf2;  // input file buffer
   int             iLenHeader;
   char            szTmp[256];
   unsigned short  iInterface;      // interface type
   char            szBandPrefix[32];// band prefix for base band unit
   int             iNumRead;

   // check for valid session handle
   if (instrSession == INVALID_SESSION) return VI_FALSE;
   // setup BB prefix
   strcpy(szBandPrefix,"");
   if (iBBPath==1) strcpy(szBandPrefix,"SOUR1:");
   if (iBBPath==2) strcpy(szBandPrefix,"SOUR2:");
   if (strlen(szFile)==0 || strlen(szInstrFile)==0)
      return VI_FALSE;
   // get file size and allocate buffer
   GetFileSize (szFile, &iFileSize);
   if (iFileSize==0)
      return VI_FALSE;
   // set default directory for waveform file
   if (!arbDemoLibWrite (":MMEM:MSIS 'D:'; *OPC?"))
      return VI_FALSE;
   if (!arbDemoLibWrite (":MMEM:CDIR '\\'; *OPC?"))
      return VI_FALSE;
   // read waveform file, open as binary !
   fp = fopen (szFile, "rb");
   if (fp==NULL)
      return VI_FALSE;
   // allocate large enough buffer and fill with zeros
   pcBuf = malloc (iFileSize + 256);
   if (pcBuf == NULL) {
      fclose(fp);
      return VI_FALSE;
   }
   memset (pcBuf, 0, iFileSize+256);
   // create header of GPIB command
   sprintf (pcBuf, ":MMEM:DATA 'D:\\%s',#%d%d",
      szInstrFile, (int)floor(log10(iFileSize)) + 1, iFileSize);
   iLenHeader = strlen(pcBuf);
   // read data to buffer and close file
   iNumRead = fread (&pcBuf[iLenHeader], 1, iFileSize, fp);
   fclose (fp);
   if (iNumRead != iFileSize) {
      free(pcBuf);
      return VI_FALSE;
   }
   // append line feed
   sprintf (&pcBuf[iLenHeader+iFileSize], "\n");
```

```
   // get connection type and setup SMU to use EOI line
   viGetAttribute (instrSession, VI_ATTR_INTF_TYPE, &iInterface);
   if (iInterface == VI_INTF_GPIB)
      arbDemoLibWrite (":SYST:COMM:GPIB:LTER EOI; *OPC?");
   // transfer data
   if (!arbDemoLibWriteBin (pcBuf, iFileSize+iLenHeader+2))  {
      free(pcBuf);
      if (iInterface == VI_INTF_GPIB)
         arbDemoLibWrite (":SYST:COMM:GPIB:LTER STAN; *OPC?");
      return VI_FALSE;
   }
   free (pcBuf);
   // set SMU back to standard (line feed)
   if (iInterface == VI_INTF_GPIB)
      arbDemoLibWrite (":SYST:COMM:GPIB:LTER STAN; *OPC?");
   // activate wave form
   if (bActivate) {
      // preset ARB unit
      sprintf (szTmp, "%sBB:ARB:PRES; *OPC?", szBandPrefix);
      if (!arbDemoLibWrite (szTmp)) { return VI_FALSE; }
      // select waveform in generator
      sprintf (szTmp, "%sBB:ARB:WAV:SEL '%s'; *OPC?", szBandPrefix, szInstrFile);
      if (!arbDemoLibWrite (szTmp)) { return VI_FALSE; }
      // set state to ON
      sprintf (szTmp, "%sBB:ARB:STAT ON; *OPC?",szBandPrefix);
      if (!arbDemoLibWrite (szTmp)) { return VI_FALSE; }
   }

   return VI_TRUE;
}
```

Main program: Using all of the above functions to generate a test signal

```
ViBoolean TestIt(void)
{
  int            i;
  double         dI[256], dQ[256];
  double         dRad;
  T_WV_DATA      WvData;
  unsigned char  ucFlags[256];

  // calculate waveform data, two sine waves
  for (i=0; i<256; i++)  {
    dRad  = 2.0*PI * (double)i / 256.0;
    dI[i] = 1.0 * sin(dRad + 0.0    ) + 0.0;
    dQ[i] = 1.0 * sin(dRad - PI/2.0 );
  }
  // setup flags
  memset (ucFlags, 0, 256);

  // fill waveform data structure
  WvData.szComment  = "Comment";
  WvData.iPoints    = 256;
  WvData.iClockRate = 54000000;
  WvData.dI         = dI;
  WvData.dQ         = dQ;
  WvData.dRmsOfs    = 0.0;
  WvData.dPeakOfs   = 0.0;
  WvData.ucFlags    = ucFlags;
```

```
  WvData.ucFlags    = ucFlags;

  // create file on local hard drive C:
  if (!arbDemoLib_GenerateWvFile (&WvData, "C:\\demo.wv"))
    return VI_FALSE;

  // open instrument connection
  if (!arbDemoLibOpen ("GPIB0::28::INSTR"))
    return VI_FALSE;

  // apply RF settings
  arbDemoLibWrite ("*RST");
  arbDemoLibWrite (":SOUR1:FREQ:MODE CW; *OPC?");
  arbDemoLibWrite (":SOUR1:FREQ 1 GHz; *OPC?");
  arbDemoLibWrite (":SOUR1:POW -30.0 dBm; *OPC?");
  arbDemoLibWrite (":OUTP1:STAT ON; *OPC?");

  // upload from C:\demo.wv -> d:\demo.wv on instrument
  if (!arbDemoLib_UploadWvFile("C:\\demo.wv", "D:\\demo.wv", 1, VI_TRUE))
    return VI_FALSE;

  if (!arbDemoLibClose())
    return VI_FALSE;

  return VI_TRUE;
}

/*
 * main program.
 */
int main (int argc, char *argv[])
{
  TestIt();

  MessagePopup ("Info","Done");

  return 0;
}
```

9 Literature and References

Product Brochures

PD 0758.0197.12	R&S SMU 200A
PD 5213.5074.12	R&S SMJ 100A
PD 0758.1893.12	R&S SMATE 200A

Manuals:

1007.9845.32-08-I	R&S SMU 200A Operating Manual
1403.4542.32-03	R&S SMJ 100A Operating Manual
1401.0940.39-03	R&S SMATE 200A Operating Manual

Application Notes:

[1] 1GP59	PC Software for generating FM/AM/PM signals in the ARB (www.rohde-schwarz.com/appnote/1GP59)
[2] 1MA28	IQWizard IQ-Signal Measurement and Conversion (www.rohde-schwarz.com/appnote/1MA28)
[3] 1GP60	Transfer Toolbox for Matlab (www.rohde-schwarz.com/appnote/1GP60)
[4] 1GP45	SMIQB60 Arbitrary Waveform Generator for SMIQ

Software Packages and Toolkits:

[5] WinIQSIM	Software for Calculating IQ Data for Modulation Generators
[6] MapWiz	Mapping Wizard for the Generation of Custom IQ Mappings
[7] IQWizard	IQ Signal Measurement and Conversion

10 Additional Information

This application note and the associated program 'ARBToolbox' are updated from time to time. Please visit the website www.rohde-schwarz.com/appnotes/1GP62 in order to download the latest versions.

11 Ordering information

SMU200A	Vector Signal Generator	1141.2005.02
SMU-B102	Frequency option 2.2 GHz, 1st RF path	1141.8503.02
SMU-B103	Frequency option 3 GHz, 1st RF path	1141.8603.02
SMU-B104	Frequency option 4 GHz, 1st RF path	1141.8603.02
SMU-B106	Frequency option 6 GHz, 1st RF path	1141.8803.02
SMU-B202	Frequency option 2.2 GHz, 2nd RF path	1141.9400.02
SMU-B203	Frequency option 3 GHz, 2nd RF path	1141.9500.02
SMU-B13	Baseband Main Module	1141.8003.04
SMU-B9	Baseband Generator with ARB (128 Msamples)	1161.0766.02
SMU-B10	Baseband Generator with ARB (64 Msamples)	1141.7007.02
SMU-B11	Baseband Generator with ARB (16 Msamples)	1159.8411.02
SMJ100A	Vector Signal Generator	1403.4507.02
SMJ-B103	Frequency option 3 GHz	1403.8502.02
SMJ-B106	Frequency option 6 GHz	1403.8702.02
SMJ-B13	Baseband Main Module	1403.9109.02
SMJ-B9	Baseband Generator with ARB (128 Msamples)	1404.1501.02
SMJ-B10	Baseband Generator with ARB (64 Msamples)	1403.8902.02
SMJ-B11	Baseband Generator with ARB (16 Msamples)	1403.9009.02
SMATE200A	Vector Signal Generator	1400.7005.02
SMATE-B103	Frequency option 3 GHz, 1st RF path	1401.1000.02
SMATE-B106	Frequency option 6 GHz, 1st RF path	1401.1200.02
SMATE-B203	Frequency option 3 GHz, 2nd RF path	1401.1400.02
SMATE-B206	Frequency option 6 GHz, 2nd RF path	1401.1600.02
SMATE-B13	Baseband Main Module	1401.2907.02
SMATE-B9	Baseband Generator with ARB (128 Msamples)	1404.7500.02
SMATE-B10	Baseband Generator with ARB (64 Msamples)	1401.2707.02
SMATE-B11	Baseband Generator with ARB (16 Msamples)	1401.2807.02
AFQ100A	I/Q Modulation Generator	1401.3003.02
SMATE-B10	Waveform Memory (256 Msamples)	1401.5106.02
SMATE-B11	Waveform Memory (1 Gsamples)	1401.5206.02
AMU200A	Baseband Signal Generator and Fading Simulator	1402.4090.02
AMU-B13	Baseband Main Module	1402.5500.02
AMU-B9	Baseband Generator with ARB (128 Msamples)	1402.8809.02
AMU-B10	Baseband Generator with ARB (64 Msamples)	1402.5300.02
AMU-B11	Baseband Generator with ARB (16 Msamples)	1402.5400.02

ROHDE & SCHWARZ GmbH & Co. KG · Mühldorfstraße 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