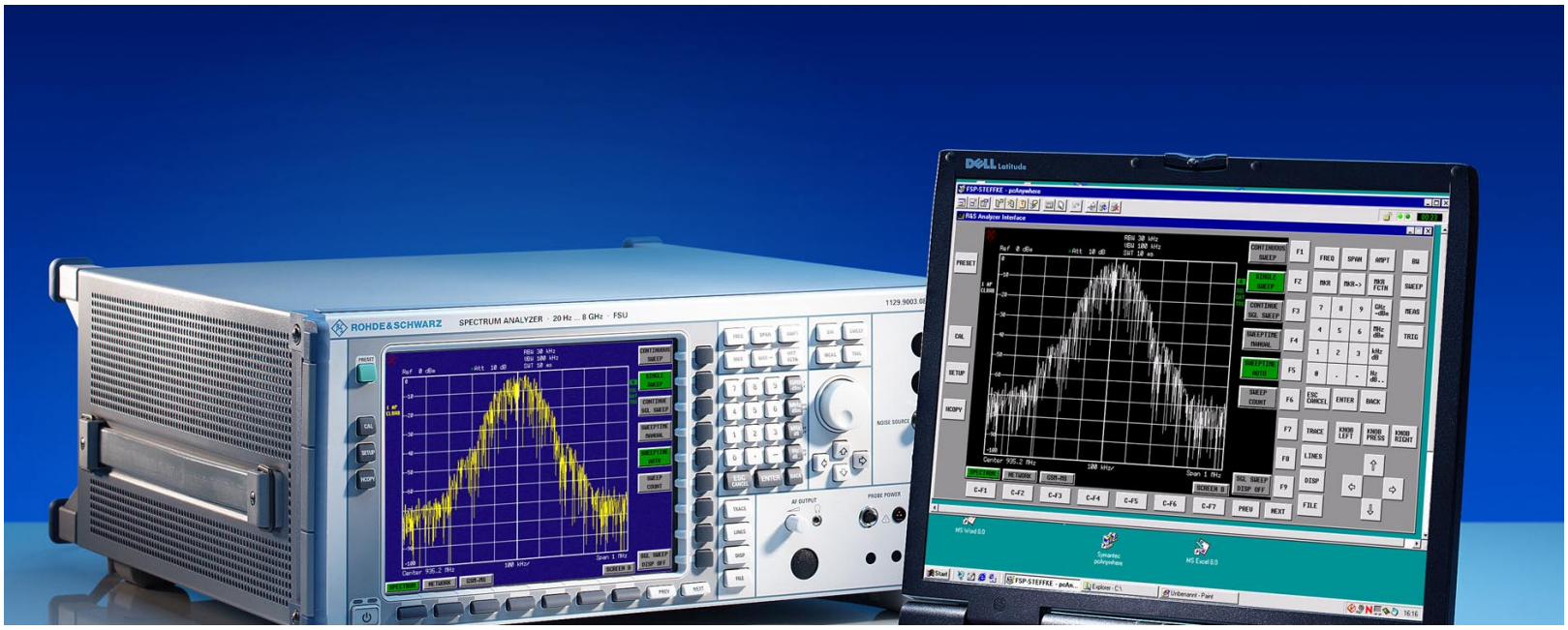

Products: FSE, FSIQ, ESIB, FSP, ESPI, FSU, ZVR, ZVC, ZVM, ZVK

Using MATLAB® for Remote Control and Data Capture with R&S Spectrum and Network Analyzers

Application Note

Spectrum and network analyzers are capable of measuring large amounts of data that require complex mathematical processing; MATLAB® is a powerful tool for such operations. This application note describes how instruments can be controlled directly from MATLAB® scripts and how measurement data can be imported into MATLAB®.

ROHDE&SCHWARZ

Subject to change – J. Ganzert 12 / 2001 – 1EF46_0E

Contents

1 Overview 2
2 Remote Control via GPIB 3
3 Remote Control via LAN 4
Installation 4
MEXRSIB Functions 4
Example Script 6
4 Remote Control using VISA 7
5 General Programming Guidelines 8
6 Programming Example 8
7 Literature 9
8 Additional Information 9

1 Overview

Spectrum and network analyzers are high-end instruments that perform various measurement tasks in today's telecommunication industry. With the increasing digital modulation, the amount of measurement data is also increasing and powerful tools for verifying the data are needed. MATLAB is a powerful tool for simulation and complex mathematical operations. Modern spectrum analyzers like FSP and FSU provide complex base band data (I/Q data) in addition to the traditional trace data. I/Q data is essential for performing analysis in the modulation domain, or using measured signals for simulation purposes.

This application note shows how to connect Rohde & Schwarz spectrum and network analyzers and MATLAB in order to perform remote control and data exchange directly from MATLAB.

2 Remote Control via GPIB

The MATLAB instrument toolbox supports communication with instruments using GPIB interface cards from several vendors such as Agilent and National Instruments.

A connection to an instrument is referred to as instrument control session. The following example shows how to establish the connection and communicate with the instrument. This example is based on National Instruments GPIB hardware and software.

1. Create the instrument object using the `gpib` function:
   ```
   <instrument object> = gpib('<vendor>', <board index>,
   <primary address>);
   ```
   e.g. `GpibInstr = gpib('ni', 0, 20);`
2. Open the connection to the instrument:
   ```
   fopen(GpibInstr);
   ```
3. Configure the connection properties, e.g. timeout, termination character, etc.:
   ```
   set(GpibInstr, 'EOSCharCode', 0);
   ```
   sets the termination character to 0 for binary data transfer.
4. Communicate with the instrument. This example reads the ID string from the instrument:
   ```
   fprintf(GpibInstr, '*IDN?');
   IDString = fscanf(GpibInstr);
   ```
5. Close the connection and clean up:
   ```
   fclose(GpibInstr);
   delete(GpibInstr);
   clear GpibInstr;
   ```

3 Remote Control via LAN

The spectrum and network analyzers can be equipped with a network interface card and thus integrated in a local area network (LAN). With the use of the RSIB interface, the instruments can be controlled over the LAN.

This approach uses an additional MATLAB DLL that provides functions for instrument control.

Installation

The RSIB interface is implemented in the RSIB32.DLL. This DLL is shipped with the instrument firmware and is located in the directory C:\R_S\INSTR\RSIB for FSE, FSIQ, ESIB, ZVx and respectively in D:\R_S\INSTR\RSIB for FSP, ESPI, FSU. The RSIB32.DLL must be copied to the directory <SystemRoot>\system32 (usually C:\WINNT\SYSTEM32) on the controller PC.

The MATLAB library MEXRSIB.DLL must be located in the working directory of MATLAB. The DLL is attached to this application note.

MEXRSIB Functions

The MEXRSIB.DLL is a MATLAB specific wrapper for the functions in the RSIB32.DLL. The following function are supported:

- `ud = mexrsib('ibfind', 'ipAddr')`
  Parameters:

`ud`	Device handle
`ipAddr`	IP address of instrument

  The function `ibfind` opens a connection to the instrument with the specified IP address. If no connection could be established, the device handle is –1.

- `mexrsib('ibwrt', ud, 'szCmdString')`
  Parameters:

`ud`	Device handle
`'szCmdString'`	Zero terminated command string

  The function `ibwrt` sends the specified command string to the instrument.

- `mexrsib('ilwrt', ud, 'szCmdString', uCount)`
  Parameters:

`ud`	Device handle
`'szCmdString'`	Zero terminated command string
`uCount`	Number of bytes to be sent

  The function `ilwrt` sends the specified number of bytes of the command string to the instrument.

- `mexrsib('ibwrtf', ud, 'fileName')`
  Parameters:

`ud`	Device handle
`'fileName'`	Name of file to be sent

  The function `ibwrtf` sends the specified file to the instrument.

- `szResponse = mexrsib('ibrd', ud)`
  Parameters:

`ud`	Device handle
`szResponse`	Zero terminated response string

  The function `ibrd` reads a zero terminated string from the instrument.

- `szResponse = mexrsib('ilrd', ud, uCount)`
  Parameters: `ud` Device handle
  `szResponse` Zero terminated response string
  `uCount` Number of bytes to be read

  The function `ilrd` reads the specified number of bytes from the instrument into a string.

- `binData = mexrsib('ilrd32', ud, uCount)`
  Parameters: `ud` Device handle
  `binData` Binary data (e.g. array of float values)
  `uCount` Number of bytes to be read

  The function `ilrd32` reads the specified number of bytes from the instrument into a binary buffer. This is typically an array of float values with measurement data.

- `mexrsib('ibrdf', ud, 'fileName')`
  Parameters: `ud` Device handle
  `'fileName'` Name of destination file

  The function `ibrdf` reads data from the instrument into the specified file.

- `mexrsib('ibtmo', ud, val)`
  Parameters: `ud` Device handle
  `val` Timeout value in seconds

  The function `ibtmo` sets the timeout value for the communication.

- `mexrsib('ibsre', ud, val)`
  Parameters: `ud` Device handle
  `val` 0 = local; 1 = remote

  The function `ibsre` sets the instrument state to local or remote.

- `mexrsib('ibloc', ud)`
  Parameters: `ud` Device handle

  The function `ibloc` sets the instrument to local state.

- `mexrsib('ibonl', ud)`
  Parameters: `ud` Device handle

  This function closes the connection to the instrument. The device handle is invalidated.

- `mexrsib('ibeot', ud, val)`
  Parameters: `ud` Device handle
  `val` 0 = no END message;
  1 = send END message

  The function `ibeot` enables the END message after write operations or disables it.

- `stb = mexrsib('ibrsp', ud)`
  Parameters: `ud` Device handle
  `stb` Status byte

  The function `ibrsp` performs a serial poll and returns the status byte.

- `srq = mexrsib('TestSrq', ud)`
  Parameters: `ud` Device handle
  `srq` 0 = no SRQ;
  1 = SRQ set

  The function `TestSrq` tests the service request (SRQ) status of the instrument.

- `srq = mexrsib('WaitSrq', ud)`
  Parameters: `ud` Device handle
  `srq` 0 = no SRQ, timeout expired;
  1 = SRQ occured

  The function `WaitSrq` waits for a service request (SRQ) of the instrument.

Example Script

The following script demonstrates a simple communication with a spectrum analyzer at IP address 89.10.66.41.

```
% Open connection to instrument with IP address
89.10.66.41
ud=mexrsib('ibfind', '89.10.66.41')

% Check for valid device handle
if ud > 0

    % Set timeout to 10 s
    mexrsib('ibtmo',ud,10)

    % Init device
    mexrsib('ibwrt',ud,'*RST;*CLS')

      % Identify the instrument
    mexrsib('ibwrt',ud,'*IDN?')
    IdStr = mexrsib('ilrd',ud,100)

    % Turn display update ON
    mexrsib('ibwrt',ud,'SYST:DISP:UPD ON')

    % Select single sweep
    mexrsib('ibwrt',ud,'INIT:CONT OFF')

    % Perform one sweep and wait for operation to
complete
    mexrsib('ibwrt',ud,'INIT;*OPC?')
    y=mexrsib('ilrd',ud,1)

    % Select binary data transfer
    mexrsib('ibwrt',ud,'FORM:DATA REAL,32')

    % Read trace data
    mexrsib('ibwrt',ud,'TRACE? TRACE1')

    % Read trace data header
    y=mexrsib('ilrd',ud,2)
    y=mexrsib('ilrd',ud,4)

    % ... and now the measurement data
    x=mexrsib('ilrd32',ud,2500/4);
    plot(x);grid;

    % Set instrument to local
    mexrsib('ibloc',ud)

end
```

4 Remote Control using VISA

VISA is a standardized software interface library providing input and output functions to communicate with measuring instruments. An initialization function in the relevant application program defines which device interface (LAN, GPIB or RS-232) is to be used. The subsequent program code is completely independent of the I/O interface.

The MATLAB instrument toolbox offers functionality to access both NI-VISA and Agilent VISA. NI-VISA version 2.5 implements a passport mechanism that extends the supported I/O channels. The following paragraphs describe how to install a passport for the RSIB interface.

Similar to communication via GPIB, a connection to an instrument is referred to as an instrument control session. The following example shows how to establish the connection and communicate with the instrument. This example is based on National Instruments GPIB hardware and software.

The MATLAB instrument toolbox and NI-VISA must be installed on the controller PC.

1. Create the instrument object using the `visa` function:
   `<instrument object> = visa('<vendor>', <resource descriptor>);`
   e.g. `viInstr = visa('ni', 'GPIB0::20::INSTR');`
2. Open the connection to the instrument:
   `fopen(viInstr);`
3. Configure the connection properties, e.g. timeout, termination character, etc.:
   `set(viInstr, 'EOSCharCode', 0);`
   sets the termination character to 0 for binary data transfer.
4. Communicate with the instrument. This example reads the ID string from the instrument:
   ```
   fprintf(viInstr, '*IDN?');
   IDString = fscanf(viInstr);
   ```
5. Close the connection and clean up:
   ```
   fclose(viInstr);
   delete(viInstr);
   clear viInstr;
   ```

The main advantage of using VISA compared to GPIB is that different I/O channels can be used without modifying the program code except for the resource descriptor.

5 General Programming Guidelines

This section mentions some general programming guidelines that apply to all remote control solutions.

Acquiring measurement data often requires a special setup of the instrument. This setup is closely related to the time required for data acquisition. When a remote control program retrieves measurement data from the instrument, it is essential that data acquisition is complete before data is read from the instrument.

Synchronization can be achieved using one of the following three methods:

- Appending “*WAI” to a remote control command forces the control program to wait at the next I/O operation until the previous command has been finished.
- When appending “*OPC?”, the subsequent read operation will return successfully only if the previous operation has been completed.
- Sending the common command “*OPC” will cause the instrument to issue a service request upon completion, if the status enable mask register and the event status enable mask register are set accordingly. The client program can then wait for the service request before proceeding with data import from the instrument.

For detailed documentation of synchronization mechanisms refer to the operating manual of the instrument.

Spectrum and network analyzers can provide measurement data in two different formats: ASCII and binary. For automated test systems it is recommended to use the binary data format, because the data size is well defined and the data transfer is faster than in ASCII format.

6 Programming Example

Attached to this application note is a MATLAB script that acquires data from a digital modulated signal using the spectrum analyzer FSU. The I/Q data from the spectrum analyzer is converted into a format that can be downloaded in the arbitrary waveform generator AMIQ. In conjunction with the signal generator SMIQ, the measured data can then be reapplied to a device under test.

The script illustrates the main steps of the process:

1. Setup the spectrum analyzer for acquisition of IQ data.
2. Acquire IQ data and transfer to the controller PC.
3. Convert IQ data for waveform generator AMIQ.

Detailed comments are included in the MATLAB script FSU_AMIQ.m

7 Literature

For more information about the RSIB interface as well as for documentation of the remote control commands, refer to the operating manual of the instrument.

For a detailed documentation of MATLAB functions and the instrument toolbox refer to the online help in MATLAB.

8 Additional Information

Please contact **TM-Applications@rsd.rohde-schwarz.com** for comments and further suggestions.

ROHDE & SCHWARZ GmbH & Co. KG · Mühldorfstraße 15 · D-81671 München · P.O.B 80 14 69 · D-81614 München ·
Telephone +49 89 4129 -0 · Fax +49 89 4129 - 13777 · Internet: http://www.rohde-schwarz.com