Accessing Measurement Data and Controlling the Vector Network Analyzer via DDE

Application Note 1EZ33_0L

Subject to change

28 April 1997, Johannes Ganzert

Products:

ZVR

ZVRE

ZVRL

ROHDE&SCHWARZ

1 Overview

The RSIB interface enables the network analyzers of the ZVR family to be controlled by means of Windows applications via DDE. The interface functions are contained in the DLL `RSIB.DLL`. The other DLL `RSDDE.DLL` provides functions for the DDE access to the instrument firmware. These functions are used by `RSIB.DLL`. The interface of these functions greatly corresponds to that of National Instruments for programming the GPIB. The function names are similar to those of the NI library but preceded by `RSDLL`. The two DLLs are part of the firmware and are updated with the firmware update kits.

The following table gives an overview of the available functions:

Function	Description
`RSDLLibfind()`	Returns a handle to device
`RSDLLibwrt()`	Writes null terminated string to device
`RSDLLibrd()`	Reads string from device
`RSDLLilwrt()`	Writes count of bytes to device
`RSDLLilrd()`	Reads count of bytes from device
`RSDLLilwrtf()`	Sends contents of file to device
`RSDLLilrdf()`	Reads data from device into file.
`RSDLLTestSrq()`	Tests for service request
`RSDLLWaitSrq()`	Waits until device issues a service request
`RSDLLibtmo()`	Sets timeout limit for device
`RSDLLibsre()`	Sets device to remote/local
`RSDLLibloc()`	Sets device temporary to local
`RSDLLibeot()`	Disable/enable END message at write operations

2 Installation and Configuration

By default the DLLs can be found in the directory `C:\RSIB`.

Copy `RSIB.DLL` and `RSDDE.DLL` to the Windows directory or to the applications directory.

3 RSIB Programming Interface

The following section describes all functions contained in `RSIB.DLL` for use in control applications. On the instrument´s hard disk are files that contain the declarations of the DLL functions and the definition of error codes for the various programming languages.

Visual Basic: `"RSIB.BAS" (D:\RUNTIME\RSIB)`

C: `"RSIBC.H" (D:\RUNTIME\RSIB)`

Winword: `"RSIBWB.BAS"(D:\RUNTIME\RSIB)`

Similar to the National Instruments interface command execution can be checked by means of the variables `ibsta`, `iberr` and `ibcntl`. Therefore all functions use references to these variables. The status word `ibsta` is returned merely as a function value by all but one function.

Status word - ibsta

All functions return a status word which contains information about the state of the RSIB interface. The following bits are used:

Mnemonic	Bit Pos.	Hex value	Description
ERR	15	8000	Function terminated with an error. If this bit is set, a more specific error code is set in `iberr`.
TIMO	14	4000	Timeout occurred during function execution. The following cases may cause this error: During wait for SRQ with the function `RSDLLWaitSrq()`. No acknowledge is received after sending data with the functions `RSDLLibwrt()` or `RSDLLilwrt()`. No answer from device after a data request with `RSDLLibrd()` or `RSDLLilrd()`.
CMPL	8	0100	Set if the answer to a data request was completely read. If data is read with `RSDLLilrd()` and the answer exceeds count of bytes the bit is cleared.

Error variable - iberr

If the ERR bit (8000h) is set in the status word, then `iberr` contains an error code which specifies the error more precisely. The RSIB has its own error codes independent of the NI interface.

Error	Error code	Description
IBERR_DEVICE_REGISTER	1	RSIB.DLL cannot register any new device.
IBERR_CONNECT	2	The connection to the device failed.
IBERR_NO_DEVICE	3	An interface function has been called with an invalid device handle.
IBERR_MEM	4	No free memory available.
IBERR_TIMEOUT	5	Timeout has occurred.

Count variable - ibcntl

The variable ibcntl is updated with the number of bytes transmitted on every read or write function call.

RSDLLibeot

VB format: `Function RSDLLibeot (ByVal ud%, ByVal v%, ibsta%, iberr%, ibcntl&) As Integer`

C format: `void FAR PASCAL RSDLLibeot( short ud, short v, short far *ibsta, short far *iberr, unsigned long far *ibcntl)`

Description: Disable/enable END message after write operations.

Parameters:

ud — The handle ud specifies a device that has been determined with the function `RSDLLibfind()`.

v — 0 - no END message

1 - send END message

Notes: If the END message is disabled, data of one command can be sent with multiple calls to write functions. The END message must be enabled before sending the last data block.

Example: See `RSDLLibwrtf()`

RSDLLibfind()

VB format: `Function RSDLLibfind (ByVal udName$, ibsta%, iberr%, ibcntl&) As Integer`

C format: `short FAR PASCAL RSDLLibfind( char far *udName, short far *ibsta, short far *iberr, unsigned long far *ibcntl)`

Description: The function returns a handle for the device named `udName`.

Parameters: udName Name of the device

Notes: This function must be called first of all.

The return value is a handle which is used by all other functions in order to address the device. If the device with name `udName` is not found the return value is negative.

For the DDE-Interface the device the name `"@local"` will be used.

Example: See `RSDLLibwrt()`

RSDLLibloc

VB format: `Function RSDLLibloc (ByVal ud%, ibsta%, iberr%, ibcntl&) As Integer`

C format: `void FAR PASCAL RSDLLibloc( short ud, short far *ibsta, short far *iberr, unsigned long far *ibcntl)`

Description: Switches the instrument temporarily to local state.

Parameter: ud The handle ud specifies a device that has been determined with the function `RSDLLibfind()`.

Notes: If the instrument was in the remote state prior to the call of this function it will be switched back to the remote state on the next access of the instrument by any other function of RSIB.DLL. The function is irrelevant in the local state.

RSDLLibrd()

VB format: `Function RSDLLibrd (ByVal ud%, ByVal Rd$, ibsta%, iberr%, ibcntl&) As Integer`

C format: `short FAR PASCAL RSDLLibrd( short ud, char far *Rd, short far *ibsta, short far *iberr, unsigned long far *ibcntl )`

Description: Reads data into the string `Rd` from the instrument specified by the handle ud.

Parameters: ud The handle ud specifies a device that has been determined with the function `RSDLLibfind()`.

Rd String into which the read data are copied.

Notes: This function fetches replies of the IEC/IEEE-bus parser to a query command. When programming in Visual Basic a string of sufficient length must be generated before calling this function. This can be accomplished with a string definition or by using the command `Space$()`.

Generating a string of length 100:

- `Dim Rd as String * 100`
- ```
  Dim Rd as String
  Rd = Space$(100)
  ```

Example: See `RSDLLibwrt()`

RSDLLibrdf()

VB format: `Function RSDLLibrdf (ByVal ud%, ByVal file$, ibsta%, iberr%, ibcntl&) As Integer`

C format: `short FAR PASCAL RSDLLibrd( short ud, char far *file, short far *ibsta, short far *iberr, unsigned long far *ibcntl )`

Description: Reads data from device with handle ud into file.

Parameters:

ud — The handle ud specifies a device that has been determined with the function `RSDLLibfind()`.

file — File that stores the data which have been read.

Notes: This functions allows reading of data with size greater than 64Kb from device.

The filename may contain drive and path.

Example: See `RSDLLibwrtf()`

RSDLLibwrt

VB format: `Function RSDLLibwrt (ByVal ud%, ByVal Wrt$, ibsta%, iberr%, ibcntl&) As Integer`

C format: `short FAR PASCAL RSDLLibwrt( short ud, char far *Wrt, short far *ibsta, short far *iberr, unsigned long far *ibcntl )`

Description: Sends commands and data to the instrument specified by handle ud.

Parameters:

ud — The handle ud specifies a device that has been determined with the function `RSDLLibfind()`.

`Wrt` — String sent to the instrument.

Notes: Setting and query commands can be sent to the IEC/IEEE-bus parser of the instrument with the function `RSDLLibwrt()`. The string parameter Wrt must be null terminated. The function automatically appends an EOS byte (0Ah) to the string.

Example: In the following Visual Basic example the start frequency of the instrument will be queried.

```
Dim ibsta As Integer    ' status variable
Dim iberr As Integer    ' error variable
Dim ibcntl As Long      ' count variable
Dim ud As Integer       ' Unit Descriptor (handle) for the instrument
Dim Cmd As String       ' command string
Dim Response As String  ' response string

' Set up link to network analyzer
ud = RSDLLibfind("@local", ibsta, iberr, ibcntl)
If (ud < 0) Then
  ' no connection established
  ' place error handling here
End If
```
```

```
' send query command to the instrument
Cmd = "SENS:FREQ:STAR?"
If (RSDLLibwrt(ud, Cmd, ibsta, iberr, ibcntl) And IBSTA_ERR) Then
  ' place error handling here
End If

' allocate space for response string
Response = Space$(100)

' query instrument reply
If (RSDLLibrd(ud, Response, ibsta, iberr, ibcntl) And IBSTA_ERR) Then
  ' place error handling here
End If
```

RSDLLibwrtf

VB format: `Function RSDLLibwrtf (ByVal ud%, ByVal file$, ibsta%, iberr%, ibcntl&) As Integer`

C format: `short FAR PASCAL RSDLLibwrt( short ud, char far *Wrt, short far *ibsta, short far *iberr, unsigned long far *ibcntl )`

Description: Sends contents of file to device with handle ud.

Parameters:

ud — The handle ud specifies a device that has been determined with the function `RSDLLibfind()`.

file — File whose contents is to be written to device.

Notes: The function `RSDLLibwrt()` allows setup and query commands to be send to the IEC/IEEE- bus parser of the instrument.

Whether data are interpreted as complete commands may be set up with the function `RSDLLibeot()` .

Example: The following example performs a save/recall via DDE

```
Dim ibsta As Integer    ' status variable
Dim iberr As Integer    ' error variable
Dim ibcntl As Long      ' count variable
Dim ud As Integer       ' Unit Descriptor (handle) for the instrument
Dim Cmd As String       ' command string

' Set up link to network analyzer
ud = RSDLLibfind("@local", ibsta, iberr, ibcntl)
If (ud < 0) Then
  ' error handling
End If

' Get setup from device
Cmd = "SYST:SET?"
RSDLLibwrt(ud, Cmd, ibsta, iberr, ibcntl) And IBSTA_ERR

' write response to file
RSDLLibrdf(ud, "C:\db.sav", ibsta, iberr, ibcntl)

' reset device
RSDLLibwrt(ud, "*RST", ibsta, iberr, ibcntl)

' restore saved setup
' disable END message
RSDLLibeot(ud, 0, ibsta, iberr, ibcntl) And IBSTA_ERR
' send command
RSDLLibwrt(ud, "SYST:SET ", ibsta, iberr, ibcntl) And IBSTA_ERR
' enable END message
RSDLLibeot(ud, 1, ibsta, iberr, ibcntl) And IBSTA_ERR
' send data
RSDLLibwrtf(ud, "C:\db.sav", ibsta, iberr, ibcntl)
```

RSDLLibsre

VB Format: `Function RSDLLibsre (ByVal ud%, ByVal v%, ibsta%, iberr%, ibcntl&) As Integer`

C format: `void FAR PASCAL RSDLLibsre( short ud, short v, short far *ibsta, short far *iberr, unsigned long far *ibcntl)`

Description: This function switches the instrument to local/remote.

Parameters:

ud	The handle ud specifies a device that has been determined with the function `RSDLLibfind()` .
v	Instrument status 0 - local 1 - remote

RSDLLibtmo

VB format: `Function RSDLLibtmo (ByVal ud%, ByVal tmo%, ibsta%, iberr%, ibcntl&) As Integer`

C format: `void FAR PASCAL RSDLLibtmo( short ud, short tmo, short far *ibsta, short far *iberr, unsigned long far *ibcntl )`

Description: Sets the timeout limit for the device.

Parameters:

ud	The handle ud specifies a device that has been determined with the function `RSDLLibfind()` .
`tmo`	Timeout time in seconds.

Notes: A timeout may occur in the following cases:

- During wait for SRQ with the function `RSDLLWaitSrq()` .
- During wait for acknowledge to a command, sent with `RSDLLibwrt()` or `RSDLLilwrt()` .
- During wait for data, which were requested by `RSDLLibrd()` or `RSDLLilrd()` .

The default value is 5 seconds.

Example: See `RSDLLWaitSRQ()`

RSDLLilrd

VB format: `Function RSDLLilrd (ByVal ud%, ByVal Rd$, ByVal Cnt&, ibsta%, iberr%, ibcntl&) As Integer`

C format: `short FAR PASCAL RSDLLilrd( short ud, char far *Rd, unsigned long Cnt, short far *ibsta, short far *iberr, unsigned long far *ibcntl )`

Description: Reads `Cnt` bytes from device with handle `ud`.

Parameters:

ud	The handle ud specifies a device that has been determined with the function `RSDLLibfind()` .
cnt	Maximum number of bytes copied into the string Rd by DLL function.

Notes: The function works similar to `RSDLLibrd()` except that `Cnt` number of bytes to be read into Rd can be explicitly specified using `RSDLLilrd()` . With this function the writing beyond the string end can be avoided with this function.

The bytes beyond count `Cnt` are lost.

Example: See `RSDLLWaitSRQ()`

RSDLLilwrt

VB format: `Function RSDLLilwrt (ByVal ud%, ByVal Wrt$, ByVal Cnt&, ibsta%, iberr%, ibcntl&) As Integer`

C format: `short FAR PASCAL RSDLLilwrt( short ud, char far *Wrt, unsigned long Cnt, short far *ibsta, short far *iberr, unsigned long far *ibcntl)`

Description: Sends `Cnt` bytes to device with handle `ud`.

Parameters:

ud	The handle ud specifies a device that has been determined with the function `RSDLLibfind()` .
`Wrt`	String sent to the device.
`Cnt`	Number of bytes sent to the device.

Notes: The function sends similar to `RSDLLibwrt()` data to a device. The end of data transmission is determined by `Cnt` and not by the null termination of the string. Therefore binary data containing null bytes can be transferred using this function. If the string is to be terminated with EOS (0Ah), this EOS byte must be appended to the string.

Example: See `RSDLLWaitSRQ()`

RSDLLTestSRQ

VB format: `Function RSDLLTestSrq (ByVal ud%, Result%, ibsta%, iberr%, ibcntl&) As Integer`

C format: `void FAR PASCAL RSDLLTestSrq( short ud, short far *result, short far *ibsta, short far *iberr, unsigned long far *ibcntl)`

Description: Checks the status of the SRQ bit.

Parameters:

ud	The handle ud specifies a device that has been determined with the function RSDLLibfind() .
result	Reference to an integer variable wherein the DLL returns the status of the SRQ bit. 0 - no SRQ. 1 - SRQ set, device issued a service request.

Notes: The function returns immediately with the current value of the SRQ bit.

RSDLLWaitSrq

VB format:
```
Function RSDLLWaitSrq (ByVal ud%, Result%, ibsta%, iberr%,
ibcntl&) As Integer
```

C format:
```
void FAR PASCAL RSDLLWaitSrq( short ud, short far *result, short
far *ibsta, short far *iberr, unsigned long far *ibcntl)
```

Description: The function waits until the device with handle ud issues a SRQ.

Parameters:

ud	The handle ud specifies a device that has been determined with the function RSDLLibfind() .
result	Reference to an integer variable wherein the DLL returns the status of the SRQ bit. 0 - no SRQ occurred within the timeout limit. 1 - SRQ set, device issued a service request.

Notes: The function waits until one of the two following events occurs:

- the device issues a SRQ
- the timeout limit set with RSDLLibtmo() expires and no SRQ occurs

Example: In the following C sample program a single sweep is started on the instrument and a marker is set to the maximum level. Before the maximum search can be performed the sweep must have finished. The synchronization is accomplished using the command "*OPC" (Operation Complete). The application waits for the SRQ using RSDLLWaitSrq(). Afterwards the marker search function is performed ("CALC:MARK:FUNC:MAX") and the response value read ("CALC:MARK:FUNC:RESULT?").

```
#define MAX_RESP_LEN        100

short               ibsta, iberr;
unsigned long       ibcntl;
short               ud;
short               srq;
char                MaxPegel[MAX_RESP_LEN];

// get handle for instrument
ud = RSDLLibfind( "@local", &ibsta, &iberr, &ibcntl );

// set timeout for RSDLLWaitSrq() to 10 seconds
RSDLLibtmo( ud, 10, &ibsta, &iberr, &ibcntl );

// if connection to instrument valid
if ( ud >= 0 ) {

  // activate SRQ by setting Event-Status-Register (ESR)
  // and enable ESB -bit within SRE register
  RSDLLibwrt( ud, "*ESE 1;*SRE 32", &ibsta, &iberr, &ibcntl );

  // set instrument to single sweep mode, start sweep and generate with
  // "*OPC" a service request after sweep has been completed
  RSDLLibwrt( ud, "INIT:CONT off;INIT;*OPC", &ibsta, &iberr, &ibcntl );

  // wait for SRQ (at sweep end)
  RSDLLWaitSrq( ud, &srq, &ibsta, &iberr, &ibcntl );
```

```
  // if sweep finished (normal operation)
  if (srq) {

     // set marker to maximum and read response (level)
     RSDLLibwrt( ud, "CALC:MARK:FUNC:MAX", &ibsta, &iberr, &ibcntl );
     RSDLLibwrt( ud, "CALC:MARK:FUNC:RESULT?", &ibsta, &iberr, &ibcntl );
     RSDLLilrd( ud, MaxPegel, MAX_RESP_LEN, &ibsta, &iberr, &ibcntl );
}
  else {
    ; // timeout handling
  }
}
else {
  ; // error: device not found
}
}
```

4 Programming Hints

4.1 General

Data length: With `RSDLLibrd()` data up to 10000 bytes may be read. The function `RSDLLilrd()` has no limitations on the data length.

4.2 Visual Basic

- Accessing the RSIB.DLL functions

In order to generate Visual Basic control applications using the DDE interface, the file `RSIB.BAS` `(D:\RUNTIME\RSIB)` should be added to the project to use the functions from RSIB.DLL.

- Declaration of the DLL functions as p rocedures

All functions return the integer value ibsta. They are therefore declared in RSIB.BAS as follows:

```
Declare Function RSDLLxxx Lib "rsib.dll" ( ... )   as Integer
```

However ibsta is also returned by reference as one of the function parameters. Therefore the functions may be declared as procedures in the following manner:

```
Declare Sub RSDLLxxx Lib "rsib.dll" ( ... )
```

- Generation of a reply buffer

Since the DLL returns null-terminated strings on replies, a string of sufficient size must be generated before - the functions `RSDLLibrd()` or `RSDLLilrd()` are called. Note that Visual Basic assigns the string a size that will not be modified by the DLL The string size can be defined with one of the following ways:

- `Dim Rd as String * 100`
- ```
  Dim Rd as String
  Rd = Space$(100)
  ```

- Reading trace data in real format Visual Basic

Replies from the instrument are always assigned to a string with the functions from `RSIB.BAS.` However reading trace binary data in real format is much faster and the processing of float values simpler than working with ASCII format. The assignment of the data to an array of float values can be done as follows:
  ```

The function declaration of RSDLLibrd() in RSIB.BAS remains unchanged:

```
Declare Function RSDLLibrd Lib "rsib.dll" (ByVal ud%, ByVal Rd$,    ibsta%,
iberr%, ibcntl&) As Integer
```

To place data directly into an array of float values, the string variable must be replaced by a suitable structure like the following:

```
Type TRACEREAL
  len As String * 6            ' Header der Real Data "#42000"
  Points(500)  As Single       ' Float-Array
End Type
```

Note: The structure must be defined in a code module.

In order to pass the structure by reference to the DLL, a special function declaration must be created:

```
Declare Function RSDLLibrdTraceReal Lib "rsib.dll" Alias "RSDLLibrd"
(ByVal ud%, rd as Any, ibsta%, iberr%, ibcntl&) As Integer
```

Using this function trace data can be read from a reply buffer of type TRACEREAL .

4.3 C / C++

- Accessing the RSIB.DLL functions

The functions of the library RSIB.DLL are declared in the header file RSIBC.H. The DLL functions can be linked to the C/C++ program in three different ways.

1. Generate the import library RSIB.LIB from RSIB.DLL using IMPLIB.EXE and add it to the project.
2. Specify the functions from RSIB.DLL in the module definition file (*.def) in the section IMPORTS.
3. Load the DLL at run time using the function LoadLibrary() and get the function pointers using GetProcAddress() . Unload RSIB.DLL from memory with FreeLibrary() before exiting the program.

In the first two cases the DLL will be loaded automatically at the startup of the application. It will be unloaded at program end provided that no other applications use it. Visual Basic for Applications use the third method in order to call functions from a DLL.

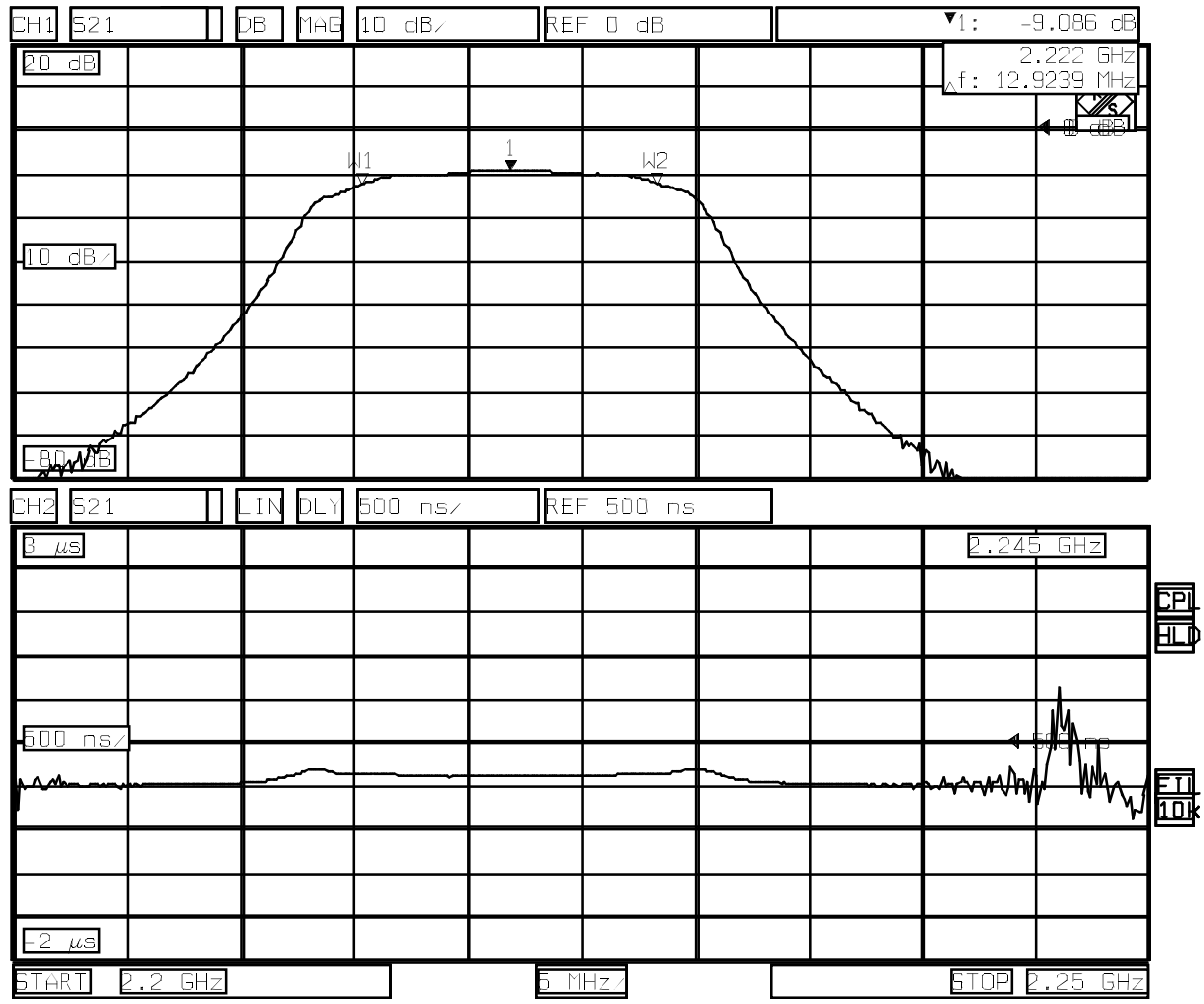
4.4 Excel

Microsoft Excel uses Visual Basic for Applications as macro language, so the functions of the library RSIB.DLL can be used in their Visual Basic format. The following sample macro performs a band-filter measurement and generates a hardcopy of the magnitude and phase of the transmission function. It uses the English versions of the object and VBA libraries (XLEN50.OLB and VBAEN.OLB). The macro has been tested with the German version of Excel 5.0 and should run without modifications in other languages since these two libraries are part of all Excel 5.0 installations. The macro is contained in the attached file ZVRDDE.XLS.

```
'
' BFilHcopy Macro
' Perform band-filter measurement
' and make hardcopy
'
'---------------------------------
' Declarations of the DLL functions
'---------------------------------
Declare Function RSDLLibfind Lib "RSIB.DLL" (ByVal udName$, ibsta%, iberr%, ibcntl&) As Integer
Declare Function RSDLLibwrt Lib "RSIB.DLL" (ByVal ud%, ByVal Wrt$, ibsta%, iberr%, ibcntl&) As
Integer
Declare Function RSDLLibrd Lib "RSIB.DLL" (ByVal ud%, ByVal Rd$, ibsta%, iberr%, ibcntl&) As
Integer
Declare Function RSDLLibwrtf Lib "RSIB.DLL" (ByVal ud%, ByVal File$, ibsta%, iberr%, ibcntl&) As
Integer
Declare Function RSDLLibrdf Lib "RSIB.DLL" (ByVal ud%, ByVal File$, ibsta%, iberr%, ibcntl&) As
Integer
```

```
Declare Function RSDLLilwrt Lib "RSIB.DLL" (ByVal ud%, ByVal Wrt$, ByVal Cnt&, ibsta%, iberr%,
ibcntl&) As Integer
Declare Function RSDLLilrd Lib "RSIB.DLL" (ByVal ud%, ByVal Rd$, ByVal Cnt&, ibsta%, iberr%,
ibcntl&) As Integer
Declare Function RSDLLTestSrq Lib "RSIB.DLL" (ByVal ud%, Result%, ibsta%, iberr%, ibcntl&) As
Integer
Declare Function RSDLLWaitSrq Lib "RSIB.DLL" (ByVal ud%, Result%, ibsta%, iberr%, ibcntl&) As
Integer
Declare Function RSDLLibtmo Lib "RSIB.DLL" (ByVal ud%, ByVal tmo%, ibsta%, iberr%, ibcntl&) As
Integer
Declare Function RSDLLibsre Lib "RSIB.DLL" (ByVal ud%, ByVal v%, ibsta%, iberr%, ibcntl&) As
Integer
Declare Function RSDLLibloc Lib "RSIB.DLL" (ByVal ud%, ibsta%, iberr%, ibcntl&) As Integer
Declare Function RSDLLibeot Lib "RSIB.DLL" (ByVal ud%, ByVal v%, ibsta%, iberr%, ibcntl&) As
Integer

Declare Sub RSDLLibwrts Lib "RSIB.DLL" (ByVal ud%, ByVal Wrt$, ibsta%, iberr%, ibcntl&)

'-------------------------------------------------
' Definitions of the bits within status word ibsta
'-------------------------------------------------
Global Const IBSTA_ERR = &h8000
Global Const IBSTA_TIMO = &h4000
Global Const IBSTA_CMPL = &h100

'------------------------------
' Codes for error variable iberr
'------------------------------
Global Const IBERR_DEVICE_REGISTER = 1
Global Const IBERR_CONNECT = 2
Global Const IBERR_NO_DEVICE = 3
Global Const IBERR_MEM = 4
Global Const IBERR_TIMEOUT = 5
Global Const IBERR_BUSY = 6
Global Const IBERR_FILE = 7

Sub BFilHcopy()
  Dim ud%, status
  Dim buffer$, cmd$
  Dim ibsta%, iberr%, ibcntl&

' get handle for device
  ud% = RSDLLibfind("@local", ibsta%, iberr%, ibcntl&)
  If (ud% < 0) Then
    ' error - exit
    msgText = "Could not connect to instrument"
    msgMode = vbYes + vbCritical
    msgTitle = "RSIB-Interface"
    respo = MsgBox(msgText, msgMode, msgTitle)
  Else
  ' reset device
    cmd$ = "*RST"
  ' send SCPI command
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)

  ' set timeout to 30s
    status = RSDLLibtmo(ud%, 30, ibsta%, iberr%, ibcntl&)
  ' switch instrument to remote state
    status = RSDLLibsre(ud%, 1, ibsta%, iberr%, ibcntl&)
  ' turn display update on
    cmd$ = ":SYST:DISP:UPD ON"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)

  ' set frequency range 2.2GHz ... 2.25GHz
    cmd$ = ":SENS:FREQ:STAR 2.2GHz;:SENS:FREQ:STOP 2.25GHz"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
  ' set measured quantity to transmission forward (S21)
    cmd$ = ":SENS1:FUNC 'XFR:POW:S21'"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
  ' change display to dual split
    cmd$ = ":DISP:FORM DSPLit"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
  ' set format of channel 1 to magnitud%e and channel 2 to group delay
    cmd$ = ":CALC1:FORM MAGN;:CALC2:FORM GDEL"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
  ' perform autoscale on channel 2
    cmd$ = ":DISP:WIND2:TRAC1:Y:SCAL:AUTO ONCe"
```

```
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
  ' switch on bandfilter markers
    cmd$ = ":CALC1:MARK1 ON;:CALC1:MARK1:FUNC BFIL"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
  ' set filter search params
    cmd$ = ":CALC1:MARK1:FUNC:BWID 3dB"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
  ' set markers search function to bandpass filter mode
    cmd$ = ":CALC1:MARK1:FUNC:BWID:MODE BPASS"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
  ' switch to single sweep
    cmd$ = ":INIT:CONT OFF"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
    cmd$ = ":INIT:IMM;*WAI"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
  ' perform search
    cmd$ = ":CALC1:MARK1:SEARCH;*WAI"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)

  ' generate hardcopy in WMF format, portrait, to file c:\user\data\bfilt.wmf
    cmd$ = ":MMEM:CDIR 'C:\USER\DATA'"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
    cmd$ = ":MMEM:NAME 'BFILT.WMF'"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
    cmd$ = ":HCOP:DEV:LANG1 WMF;*WAI"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
    cmd$ = ":HCOP:PAGE:ORI PORT"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
    cmd$ = ":HCOP:DEST 'MMEM'"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
  ' start hardcopy
    cmd$ = ":HCOP:IMM;*WAI"
    status = RSDLLibwrt(ud%, cmd$, ibsta%, iberr%, ibcntl&)
  ' switch instrument back to local
    status = RSDLLibsre(ud%, 0, ibsta%, iberr%, ibcntl&)

  ' insert meta file
    ChDir "C:\USER\DATA"
    ActiveSheet.Pictures.Insert("BFILT.WMF").Select
  End If

End Sub
```



Date: 28.APR.97 15:40:23

4.5 Winword

The functions of the library `RSIB.DLL` can be accessed from Winword macros with some restrictions. The function declarations contained in the file `rsibwb.bas` must be copied into the macro.

Restrictions:

Declaring functions with the statement `Declare` allows for parameters of type integer to be passed by value only. Since the functions of the library `RSIB.DLL` expect references for the variables ibsta, iberr and ibcntl these must be declared as strings. This fact has to be taken into account when checking error codes and requesting values from the instrument.

Note that Winword 2.0 and 6.0 do not use Visual Basic for Applications, the macro therefore is always language specific.

Johannes Ganzert, 1ES1
Rohde & Schwarz
28 April 1997