Printing Tool for Audio Analyzer R&S® UPV

Application Note

Products:

- R&S®UPV
- R&S®UPV66
- R&S®UPV-K1

This application note provides a tool for printing a report about measurement results on the audio analyzer R&S® UPV. The executable can be freely configured to include any result graphs and figures into the report. In addition it is an application example for a class library which can be included into custom application programs to add a printing function.

Template: 3573.7380.02/ CI 01.00

ROHDE&SCHWARZ

Thomas Lechner

16 June 2011-0e

Table of Contents

1 Introduction

The functions of the audio analyzer R&S® UPV, further herein below called “UPV”, can be fully controlled using SCPI commands. External remote control is available via GPIB and LAN interface with option UPV-K4. For internal sequence control, option UPV-K1 provides a TCP connection to send SCPI commands from a program running on the UPV itself to the firmware via “UPxServer.dll” as local remote control host. As a programming language, Visual Basic .NET is provided with option UPV-K1.

This mechanism provides a powerful tool for creating custom measurement functionality including Windows® graphic user interface. Examples of such application programs are provided in several application notes.

When performing complex measurement sequences, it is further desirable to print protocols of measurement results or store them as electronic documents, e.g. in pdf format. As the printing functionality with VB.NET under Windows® is not as straightforward as building a user interface, this application note provides a class library plus some guidance on how to integrate printing functionality into custom sequence programs on the UPV.

As a sample application of the printing class library, a complete application program is provided which allows to configure and perform a printout of numeric and graphic results on the UPV. This program is also provided for stand-alone use as an installer.

Chapter 2 gives an overview over UPV remote commands for dealing with all kinds of results. This includes querying numeric results and traces from graphic displays as well as configuration of displays and hardcopies and writing back traces obtained by the sequence program to one of the sweep graph displays.

Chapter 3 explains the basics of printing functionality in VB.NET for printers installed in the Windows® operating system.

Chapter 4 is a shortform manual for the application program provided. This application program can be used without the background knowledge given in the two preceding chapters.

Chapter 5 is a guide for using the class library provided in custom application programs. The source code of the sample application program may be used as reference example.

2 Measurement Results on the Audio Analyzer R&S® UPV

2.1 Numeric Results

Numeric results are measurement results which consist of a single number. They comprise the main analyzer function result, input monitor result, level monitor result and frequency, frequency and phase or frequency and group delay result.

For each numeric result a combi display can be opened by double-clicking into the respective numeric field. The combi display stores a maximum and minimum reading of the numeric result and offers limit checking. The maximum and minimum readings specify the maximum or minimum value, respectively, which has occurred in continuous or repeated single measurements. They are reset by pressing the “Start” key or by the remote command “INIT:CONT ON”. Furthermore actual value, maximum and minimum can be visualized in a bar graph.

The SCPI query command for numeric results in application programs is

SENSe<n>:DATA[1]|2,3,…16?

<n> specifies the source of the result:

Query parameter <n>					
Instrument Type	Main Analyzer Function	Input Monitor	Frequency	Phase / Group Delay	Level Monitor
<n>	1	2	3	4	6

Table 1: Query parameter for numeric results

The second parameter **[1]|2,3,…16** specifies the analyzer input channel.

Maximum and minimum readings can be queried with the SCPI commands

SENSe<n>:DATA[1]|2,3,…16? MIN
SENSe<n>:DATA[1]|2,3,…16? MAX

2.2 Result Graphs

Result graphs can contain sweep traces, waveform, spectrum (FFT, 1/n octave or harmonics) or special results like MOS or delay over time (PESQ with option UPV-K61). The UPV provides several graphic windows to display this data.

2.2.1 Storing and Querying Trace Data

Trace data can be stored in ASCII format, e.g. for export to spreadsheets, from the field on the bottom of the respective graph config panel. The graph config panel is opened in the UPV window from the DispConfig menu or by right-clicking into the respective graph and choosing “Config” from the context menu.

Trace data can be queried (X values and Y values separately) by remote control and sequence control programs. The query command

TRACe:Subsys<i>:LOAD:AX|Y?

returns all X or Y values, respectively, of a trace, where “Subsys<i>” specifies the particular graph. By default the response is comma-delimited in ASCII format. Alternatively data can also be transferred in REAL format. As this format is more efficient, it is recommended for large scans like FFT spectra and waveform traces. For details see chapter “Triggering Measurements and Reading Results” in the section of the UPV manual about remote control.

2.2.2 Writing Back Trace Data to Graphs

For application programs it may be desirable to display trace data which has been calculated or collected by the application program e.g. in a sweep graph. In an example the application program reads two traces, subtracts the levels and displays the result in a new trace.

To write a sweep to one of the sweep graphs, the graph must be configured accordingly and “X Source” must be set to “Manual”. This can be done either by (SCPI) remote commands or by loading an appropriate setup. The respective SCPI command is

DISPlay:SWEep<i>:X:SOURce MAN

Trace data can be written with the command

TRACe:Subsys<i>:STORE:AX|Y

Here is sample code for writing a trace to sweep graph 1. The method “UPVCommand” sends the argument string to the UPV and queries the error state.

```
UPVCommand("DISP:SWE:X:SOUR MAN")
UPVCommand("DISP:SWE:X:SCAL MAN")
UPVCommand("DISP:SWE:X:AXIS FREQ")
UPVCommand("DISP:SWE:X:UNIT HZ")
UPVCommand("DISP:SWE:X:LEFT " & LeftValue.ToString & " HZ")
UPVCommand("DISP:SWE:X:RIGHT " & RightValue.ToString & " HZ")
UPVCommand("TRAC:SWE:STORE:AX " & SweepXString)
UPVCommand("TRAC:SWE:STORE:AY " & SweepYString)
```

X values can only be written to sweep graphs. To achieve a desired X axis scaling for a FFT graph or waveform graph, the FFT or waveform has to be configured appropriately before writing the Y data, i.e. the sample rate and FFT size or waveform length, respectively has to match that of the Y data to be written.

2.2.3 Hardcopies of Graphs

To include result graphs into printouts or reports, hardcopies can be used. Hardcopies can be generated from the UPV window as bitmap copy or widely configurable in size and appearance from the currently active graph. The hardcopy can be sent to the Windows printer, stored to a file or put on the clipboard for use as an image by another Windows program.

The hardcopy is configured in the “Config Panel” (menu item “Utilities → Config Panel).

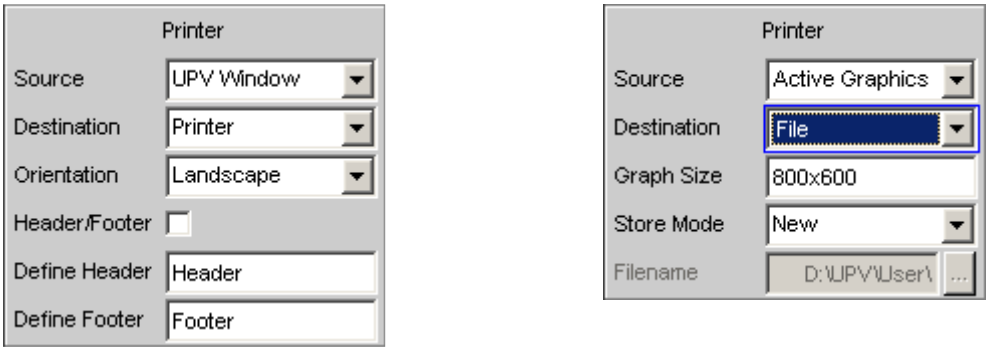

Figure 1: Configuration of the hardcopy in the config panel, left for sending the complete UPV window to a Windows printer, right for storing the active graph to a file.

ASCII commands for remotely configuring the hardcopy function can easily be logged in the SCPI recorder (menu item “Sequence → SCPI Recording”). It is initiated either by pressing the “Hcopy” key on the front panel or remotely by ASCII command

HCOPy:IMMediate

For a hardcopy of the active graph, the selected graph has to have the Windows focus (the header of the window is blue instead of grey). In the user interface this is achieved by clicking on the respective window. In remote control a possibility for getting the focus on a graph window is to open it remotely. First send a remote command to close the window for the case that it is visible:

DISPlay:SWEep:SHOW OFF

Then open the window with

DISPlay:SWEep:SHOW ON

Because this window has been treated last by the program, it will have the focus and can be hardcopied.

For details see the UPV manual and the source code of the application program.

2.2.4 Graphic Profiles

2.2.4.1 Purpose of Graphic Profiles

Formatting information for the UPV graphs can be stored in so-called graphic profiles with file extension “*.gpf”. This information includes colors for all objects, line width and styles, fonts, font size and properties etc.

2.2.4.2 Loading Graphic Profiles

Graphic profiles can be loaded in the config panel separately for screen, printer, file and clipboard. For example, the standard background on the screen is black whereas a white background is more printer-friendly.

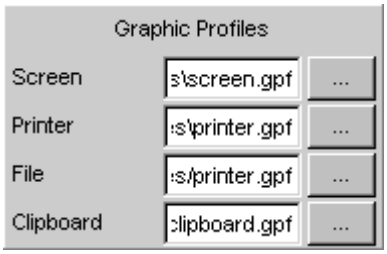

Figure 2: “Graphic Profiles” field in the UPV Config Panel

2.2.4.3 Editing and Storing Graphic Profiles

For editing or creating a graphic profile, put the focus on a graphic window (by clicking with the mouse on the header) and select menu item “Utilities → Edit Graphic Profiles …”. The graphic profile editor window opens. It shows a copy of the selected graphic window and some controls on the bottom.

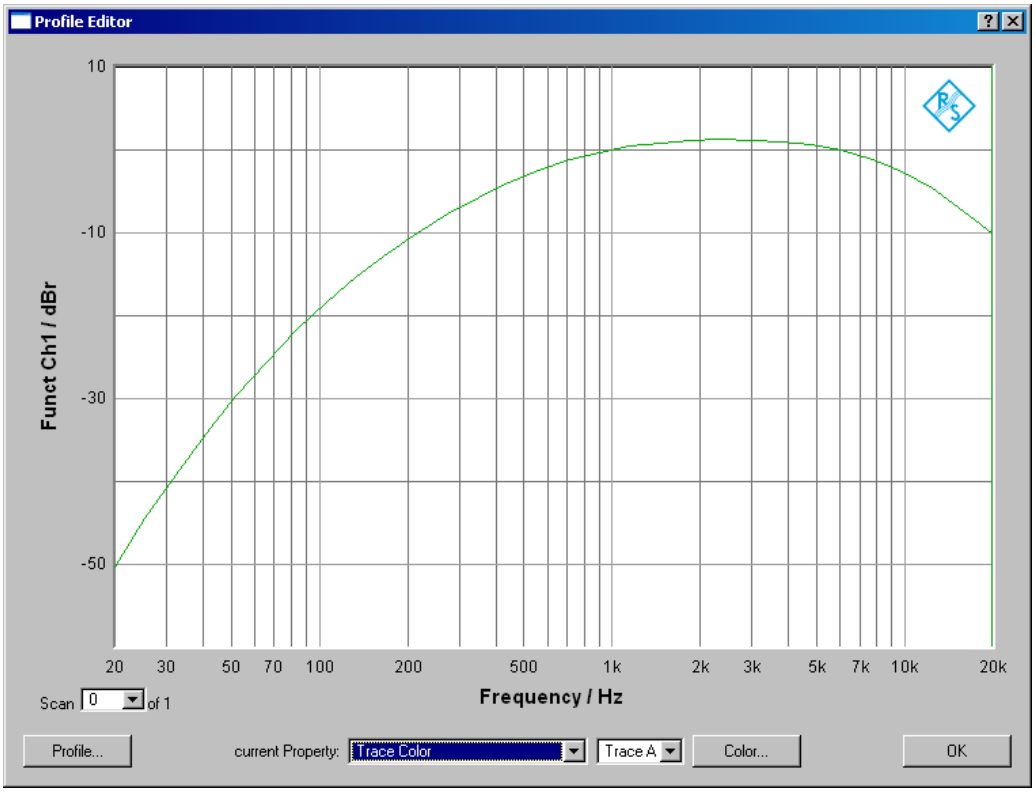

Figure 3: Graphic Profile editor

Click the “Profile …” button and load an existing profile which should be modified or specify a new file name for creating a new profile. Select one of the properties in the “Current Property” combobox and enter the desired value on the right side or click the button to open a color or font dialog. Any change will be reflected in the copy of the graphic window above the controls.

Once all settings have been done as desired, click the “Ok” button to close the window and save the current profile under the previously specified name.

3 Windows Printing Function in VB.NET

3.1 Introduction

The windows printing function sends a printout to a printer which is installed with a printer driver. Such a printer could be locally connected, a network printer or a PDF distiller which is installed as windows printer.

The Windows printing function as implemented in the VB.Net environment is object oriented. Instead of sequentially sending characters to the printer, a print document is defined. Once the printout is initiated e.g. with a print dialog control, or if a print preview control is opened, the start of each printed page raises a “PrintPage” event. The arguments of this event return a handler for the graphics of the page to be printed, which can be accessed and filled by the event handler code. The procedure for a printout is the following:

- The application program instantiates a print document
- The application program opens a print dialog or print preview control associated with the print document
- A PrintPage event is raised for each page to be printed
- The application program fills each page with content in an event handler for the PrintPage event and returns the control to the printing object once a page has been filled
- The application program must memorize the print progress from the end of one page until the start of the next page, e.g. in a global variable.

Consequently the document to be printed can only be filled when printout has been started. The contents have to be preliminarily stored from the moment of the availability of data (e.g. numeric results, screenshots) until the moment when the printout is started.

In the following the procedure will be illustrated using code from the application program.

3.2 The PrintDocument Object

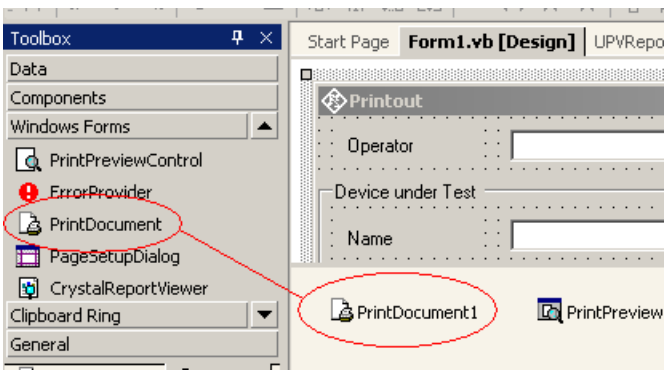

Figure 4: Inserting a PrintDocument into a Windows form

The print document is dragged from the toolbox and dropped to the designer of the respective window.

3.3 PrintDialog and PrintPreview Controls

The PrintDialog and / or PrintPreview controls are dragged from the toolbox and dropped to the designer of the respective window. It is opened in the code with

```
With PrintDialog1
    .Document = PrintDocument1     'associate with PrintDocument
    .AllowSomePages = True    'allow printing of subset of pages
    If .ShowDialog = DialogResult.OK Then   'do the following of
the Ok button is clicked
         PrintDocument1.Print()              'print the document
    End If
End With
```

The print dialog window stays open until the user either clicks “Ok” or “Cancel”.

```
With PrintPreviewDialog1
    .Document = PrintDocument1     'associate with PrintDocument
    .WindowState = FormWindowState.Maximized     'maximize print
preview window
    .ShowDialog()                      'show print preview window
End With
```

The print preview window stays open until it is closed by user interaction.

3.4 Start the printout

The printout of the document can be started without PrintDialog or PrintPreviewDialog with the “Print” method of the PrintDocument class:

```
PrintDocument1.Print()
```

Note that at this time the PrintDocument has no content. The pages are filled by the event handler for the PrintPage event.

3.5 The PrintPage Event

An event handler for the PrintPage event must be inserted into the code in which the PrintDocument is instantiated. An event handler is a method which is called when the respective event is raised somewhere else in the code.

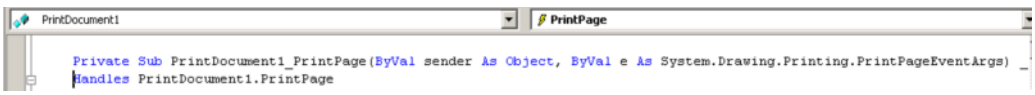

Figure 5: Event handler for the PrintPage event of a PrintDocument

e is the handle for the graphics of the page to be printed. The page can now be filled with graphics commands.

```
gr = e.Graphics       'graphics object of the page to be printed
gr.DrawString(HeadlineText, _HeadlineFont, _HeadlineBrush, _
HeadlineRectangle)      'write formatted text

gr.DrawLine(_LinePen, _PrintRect.Left, _CurrentYLocation, _
_PrintRect.Right, _CurrentYLocation)      'draw horizontal line
```

etc.

3.6 End of Page and End of Document

When the event handler terminates, the end of the page is reached and the real printout or preview display of this page starts. Before the end of the event handler the property “HasMorePages” has to be set to define whether more pages are to follow.

```
e.HasMorePages = True
```

If this property is true, the “PrintPage” event is raised again for the next page after the event handler has terminated. If the property is false, an “EndPrint” event is raised which can be used to clean up local variables of the printout process, reset page numbering etc.

4 Using the Application Program

4.1 Prerequisites

Running the application program requires option UPV-K1 “Universal Sequence Controller” to be installed. It further requires .NET framework 2.0 which is installed on all UPVs with firmware version 2.0.0 and higher. The Visual Basic .NET development environment delivered with this option is not required to run the installed application program.

4.2 Installation

Copy the installer “1GA57.msi” to the UPV hard disk, e.g. to “D:\R&S_Software\Applications. Run the installer and follow the instructions on the screen.

Shortcuts are installed on the desktop and in the programs menu in subfolder “R&S UPV Applications”.

4.3 Starting the Application Program

The program can be started from the shortcut “1GA57 Printout” which can be found both on the desktop and in subfolder “R&S UPV Applications” in the Programs menu.

It is further possible to configure one of the shortcut buttons in the toolbar to directly start the program. Use menu item “Utilities → Quick Launch Config…” in the UPV window, click the button on the right side on one of the rows, select file extension “*.exe” and browse to “C:\Program Files\Rohde&Schwarz\1GA57\Printout1.exe”.

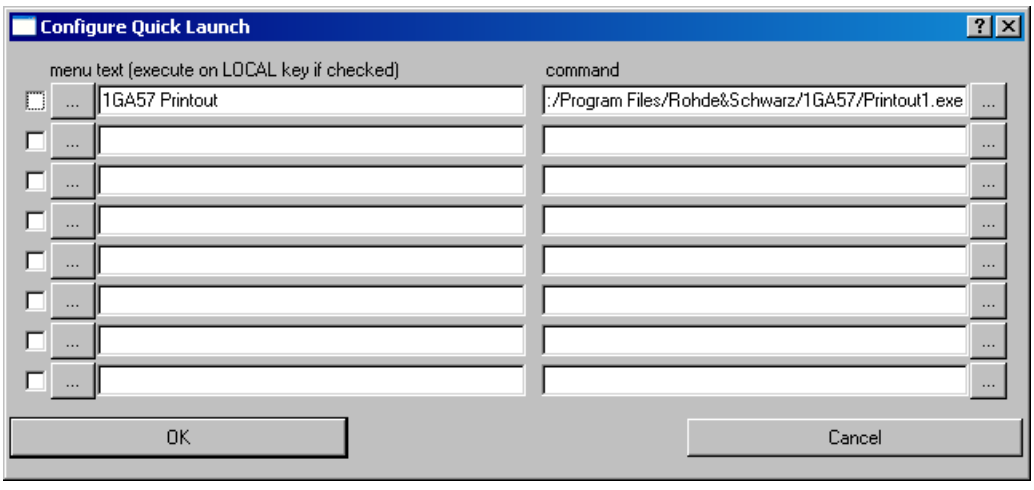

Figure 6: Quick launch button configuration panel

4.4 User Interface

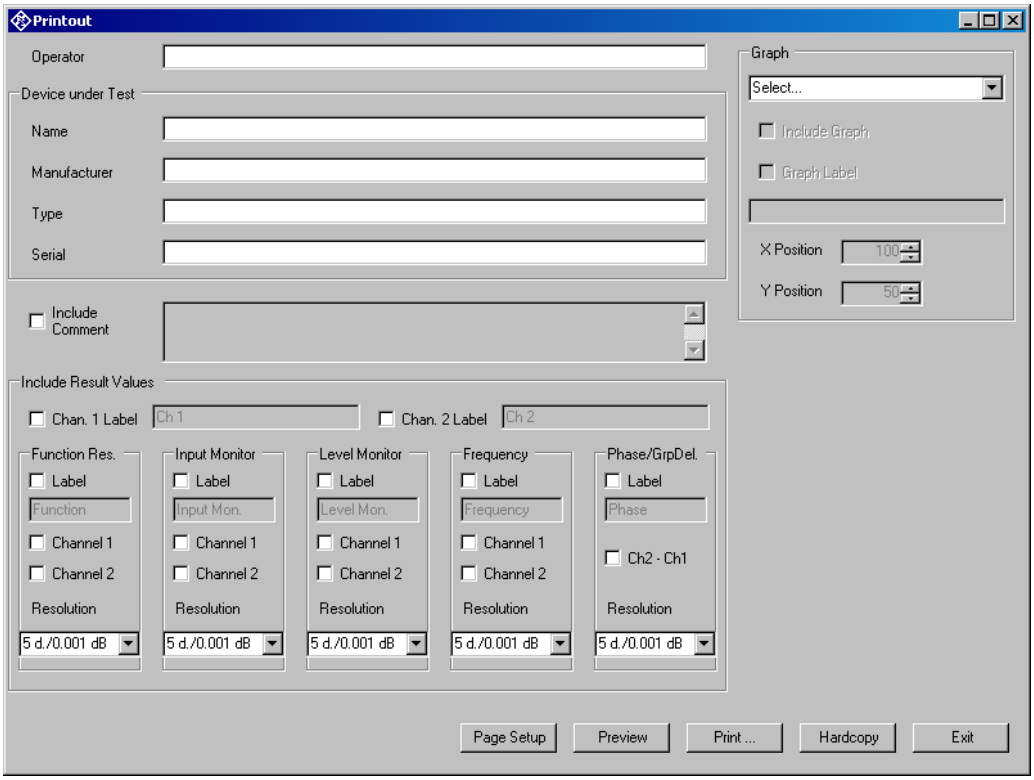

Figure 7: User interface of the application program

In the text boxes on top, information about operator and device under test can be entered, as well as a comment if desired.

The field “Include Result Values” allows to specify numeric results to be included into the report, together with labeling and formatting options. The labels are used irrespective of the functions set in the UPV.

In the “Graph” field, one or more of the available UPV graphs can be selected for inclusion into the report. If desired, a comment can be inserted into the hardcopy.

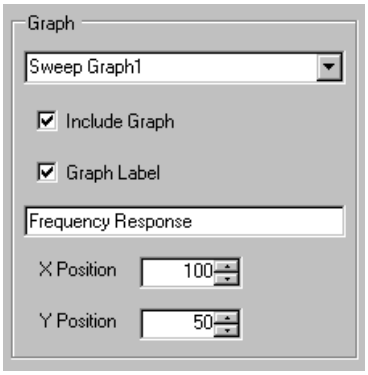

Figure 8: Graph field of the user interface

The “Page Setup” button on bottom allows to select paper format, margins etc.

The “Preview” button generates a preview of the report which can be printed from the preview button.

The “Print” button opens the usual Windows print dialog where printer, print range and number of pages can be selected.

The “Hardcopy” button starts the printout of the report without any further user interaction required.

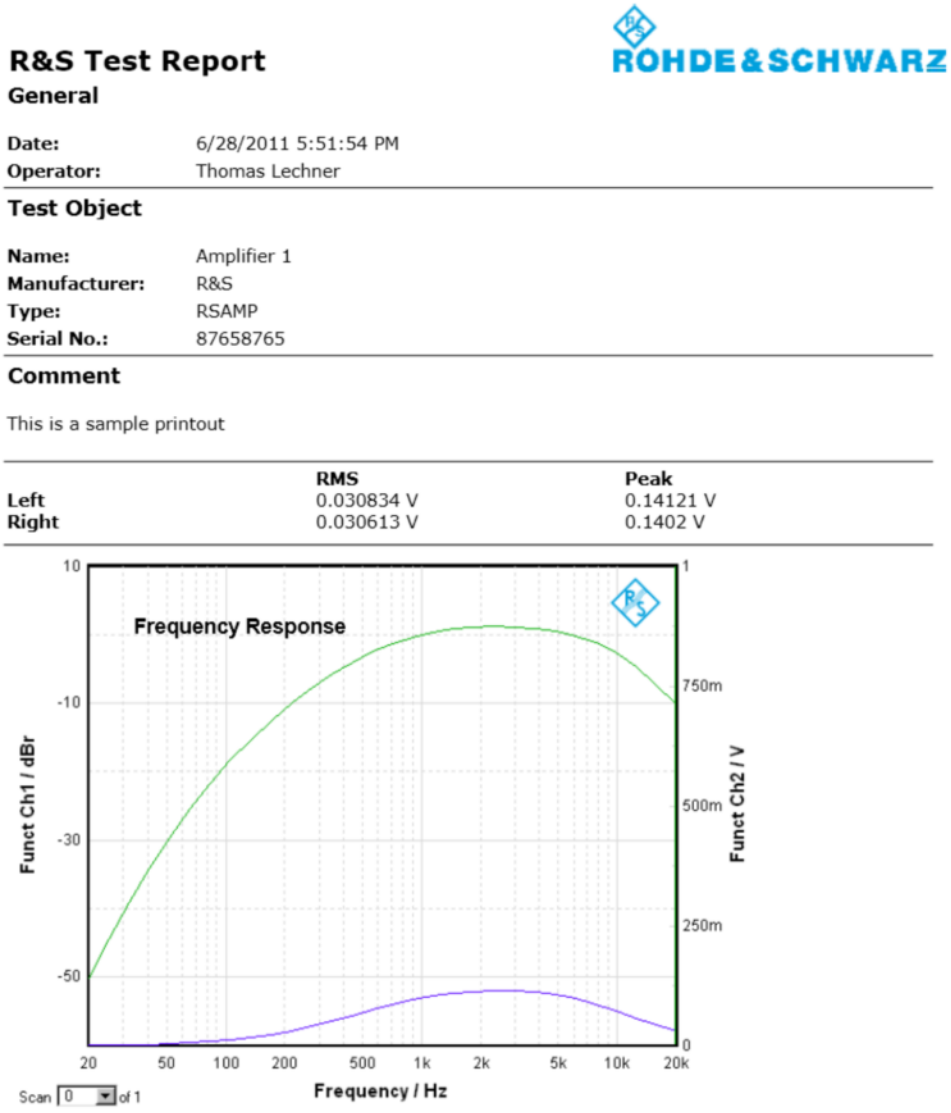

Figure 9: Sample printout of the application program

5 Working with the Source Code

5.1 Prerequisites

Editing, compiling and running application programs on the UPV requires option UPV-K1 to be installed on the instrument. The option also comprises the integrated development environment (IDE) for Visual Basic .NET. Depending on the date of shipment, there are three different versions of VB.NET: 2003, 2005 and 2008. For convenience, a complete set of project files is provided with this application note for each of the versions of the programming language.

5.2 Extracting and Storing the Project Files

Select and download the zipped source code “1GA57_VBxxxx.zip” according to version xxxx of the Visual Basic .NET programming environment installed on your UPV. Extract the archive to a folder on the UPV hard disk. It is recommended to use partition “D:”, for example “D:\Visual Studio Projects” or “D:\UPV\VBExamples”.

5.3 Opening the Solution

Use item “File → Open Project” in the programming environment menu or button “Open Project” on the start page, browse to “Printout1.sln” and open it.

To open the source code, right-click on “Form1.vb” in the Solution Explorer and click “View Code”. To edit the graphic user interface, select “View Designer” in the same context menu. The code for the printing class library is in item “UPVReport.vb”.

5.4 Running the Program in the Debugger

For running the program in the debugger, the configuration in the IDE (Integrated Development Environment) must be set to “Debug”. Start the program with menu item “Debug → Start” or with the start button in the debug toolbar. Every start in the debugger is preceded by a new compilation of the code so that modifications of the source code have an immediate effect. While the program is running in the debugger, the editor windows for the source code are locked.

6 Integration of the Class Library into Other Application Programs

6.1 Adding the Class Library to the Source Code

To add the class library to an existing project, right-click on the project name in the Solution Explorer, choose “Add” from the context menu and “Add Existing Item …” from the submenu. Browse to the file “UPVReport.vb” in the source code provided with this application note and click “Ok”. The file will be copied to the folder of the project where it is to be added.

For a detailed example for integration of the class library into an application program, please refer to the source code of “Form1.vb” of the application program provided with this application note.

6.2 Instantiating the Report Object

To use the code, an object must be instantiated from the class “UPVReport” in the global definition area of the project.

```
Private WithEvents RPT As New UPVReport
```

It is essential that the object is instantiated “WithEvents” because the events will be needed to use the class library.

The class includes the PrintDocument and all controls needed for the printing functionality. Furthermore it provides methods for filling the pages like adding logo, headline, tables and images.

6.3 Properties

The class definition “UPVReport” provides the following public properties:

```
Public Property BoldTextFont() As System.Drawing.Font
Public Property CurrentPage() As Integer
Public Property CurrentYLocation() As Integer
Public Property HasMorePages() As Boolean
Public Property HeaderWidth() As Single
Public Property HeadlineBrush() As SolidBrush
Public Property HeadlineFont() As System.Drawing.Font
Public Property HeadlineSpace() As Single
Public Property ImageSpace() As Single
Public Property LabelBrush() As SolidBrush
Public Property LabelFont() As System.Drawing.Font
Public Property LinePen() As System.Drawing.Pen
```

```
Public Property LineSpace() As Single
Public Property Logo() As System.Drawing.Image
Public Property PreSubHeadlineSpace() As Single
Public Property SubHeadlineBrush() As SolidBrush
Public Property SubHeadlineFont() As System.Drawing.Font
Public Property SubHeadlineSpace() As Single
Public Property TableSpace() As Single
Public Property TextBrush() As SolidBrush
Public Property TextFont() As System.Drawing.Font
Public Property TextSpace() As Single
```

To set or read a property, type the name of the report object followed by a dot "." And the name of the property:

```
RPT.HasMorePages = True
```

6.4 Printing functions

The following methods start page setup dialog, print dialog, print preview and immediate print of the document:

```
Public Sub PageSetup()
Public Sub PrintDialog()
Public Sub PrintPreview()
Public Sub Print()
```

The following methods can be used to enter content to the page in print:

```
Public Function AppendImage(ByVal Img As System.Drawing.Image)_
As PrintResult

Public Function AppendImage(ByVal Img As System.Drawing.Image,_
ByVal Label As String, ByVal LabelXPosition As Integer, ByVal _
LabelYPosition As Integer) As PrintResult

Public Function AppendHeadline(ByVal HeadlineText As String) _
As PrintResult

Public Function AppendSubHeadline(ByVal SubHeadlineText As _
String) As PrintResult

Public Function AppendValueLine(ByVal HeaderText As String, _
ByVal ValueText As String) As PrintResult

Public Function AppendTable(ByVal Tbl As ReportTable, ByVal _
Borders As Boolean) As PrintResult

Public Function PlaceString(ByVal Text As String, ByVal _
RelXPosition As Single, ByVal RelYPosition As Single, ByVal _
Width As Single) As PrintResult

Public Function HorizontalLine() As PrintResult
```

The return value “PrintResult” reports the success of the printing operation:

```
Public Enum PrintResult
    Success
    PageFull
    PrintError
End Enum
```

“Success” means that the information could be placed on the page as intended.
“PageFull” means that the remaining space on the page is not sufficient to place the information. The current page has to be wrapped up with “HasMorePages” set to “True”, and the information must be placed on the next page.
“PrintError” means that an unknown error occurred which prevented the method from placing the information on the current page.

“ReportTable” is an own class definition for defining and filling a table, and for handing it to the respective printing function:

```
Public Class ReportTable

    Structure TableRow
        Dim Cell() As String
    End Structure

    Public Sub New(ByVal ColumnCount As Byte, ByVal RowCount As_
    Byte, ByVal WithColumnHeader As Boolean, _
    ByVal WithRowHeader As Boolean)

    Public Property Cell(ByVal Row As Byte, ByVal Column As _
    Byte) As String

    Public Property ColumnHeader() As String()

    Public Property RowHeader() As String()

    Public Property HasRowHeader() As Boolean

    Public Property HasColumnHeader() As Boolean

    Public Property RowCount() As Byte

    Public Property ColumnCount() As Byte

End Class
```

To print a table to a page, instantiate a table object:

```
Dim ResultValues As New UPVReport.ReportTable(ColumnCount, _
RowCount, CatLabel, ChLabel)
```

Next fill the cells of the table:

```
With ResultValues
    .ColumnHeader = ColumnHeader
    .RowHeader = RowHeader
    …

End With
```

Then send the table object to the method placing it in the document:

```
Response = Print.AppendTable(ResultValues, False)
```

6.5 Events

The following events are provided by the class:

```
Public Event PrintPage()
Public Event EndPrint()
Public Event BeginPrint()
```

The event handler for the “PrintPage” event is used to fill the current page in print, using the printing functions described above.

```
Private Sub RPT_PrintPage() Handles RPT.PrintPage

    … (Variable Definitions )

    Try
        With RPT
            If CurrentPosition = 0 Then
                .AppendHeadline("R&S Test Report")
                .AppendSubHeadline("General")
                .AppendValueLine("Date:", DateTime.Now.ToString)
                .AppendValueLine("Operator:", TextBox1.Text)
                .HorizontalLine()
                .AppendSubHeadline("Test Object")
                .AppendValueLine("Name:", TextBox2.Text)
                .AppendValueLine("Manufacturer:", TextBox3.Text)
                .AppendValueLine("Type:", TextBox4.Text)
                .AppendValueLine("Serial No.:", TextBox5.Text)
                .HorizontalLine()

                  …

    Catch ex As Exception
        Debug.WriteLine(ex.ToString)
        MessageBox.Show("Error in print preparation")
    End Try
End Sub
```

7 Literature

- R&S UPV Operating Manual
- Rod Stephens: “Visual Basic 2008 Programmer's Reference”, Wrox (February 5, 2008)
- Rakesh Rajan: “Visual Basic 2008 Recipes: A Problem-Solution Approach”, Apress (April 24, 2008)

8 Ordering Information

Audio Analyzer		
Type	**Designation**	**Order No.**
UPV	Audio analyzer, analog interfaces, DC to 250 kHz	1146.2003.02
UPV66	Audio analyzer, without display, keyboard and CD drive	1146.2003.66
UPV-K1	Universal sequence controller	1401.7009.02

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

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

Certified Quality System
ISO 9001

Regional contact
Europe, Africa, Middle East
+49 89 4129 12345
customersupport@rohde-schwarz.com

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

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

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

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

www.rohde-schwarz.com