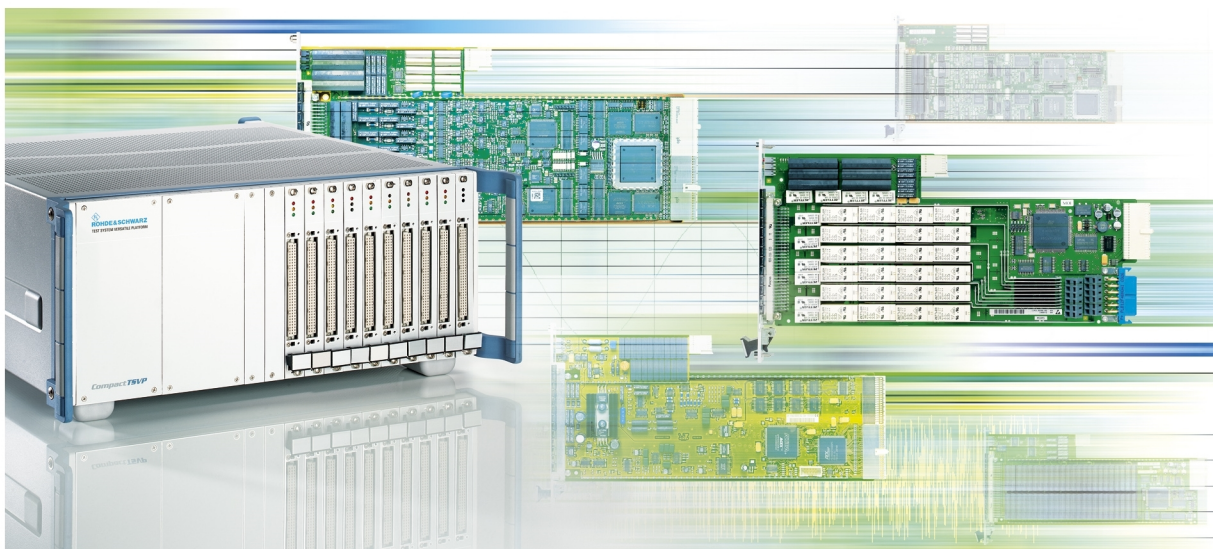# R&S®EGTSL
Enhanced Generic Test Software Library
User Manual

1143414042
Version 17

ROHDE & SCHWARZ

Make ideas real

This Software Description is valid for the following software versions.

- R&S®GTSL version 3.00 and higher versions
- R&S®EGTSL version 3.00 and higher versions

Contents

1   General

1.1   User Information

This software description applies to the following ROHDE & SCHWARZ products:

| R&S TS-LEGT | 1143.4140.02 | Enhanced R&S GTSL Software for ICT |
| R&S TS-LEG2 | 1166.3992.02 | Enhanced R&S GTSL Software for Basic ICT on R&S TS-PSAM |

1.2   Reference Documents

The Enhanced Generic Test Software Library R&S EGTSL is part of the Generic Test Software Library R&S GTSL. For this reason, the following documentation is to be noted in addition to this software description:

- Software Description Generic Test Software Library R&S GTSL

The related test hardware is required for performing in-circuit tests. The test hardware is described in the following documentation:

- User Manual Test System Versatile Platform R&S CompactTSVP
- User Manual Test System Versatile Platform R&S PowerTSVP
- User Manual Analog Source and Measurement Module R&S TS-PSAM
- User Manual ICT Extension Module R&S TS-PICT
- User Manual Matrix Module B R&S TS-PMB
- User Manual Power Supply/Load Module R&S TS-PSU
- User Manual Power Supply/Load Module 12 V R&S TS-PSU12
- Documentation of the test adapter

2 Software Installation

2.1 General

ⓘ Enhanced Generic Test Software Library R&S EGTSL is installed using the installation routine for the Generic Test Software Library R&S GTSL.

The Generic Test Software Library R&S GTSL can be downloaded from R&S GLORIS server. After extracting the compressed installation file, the whole contents of the installation can be found in the target directory. Please read the README.TXT file before starting the installation by executing the setup.exe file.

To install the Generic Test Software Library R&S GTSL under Windows 10 or Windows 7, the user must be logged in as administrator or as a user with administrator rights.

For additional information on the de-installation of previous versions of the Generic Test Software Library R&S GTSL or concerning installation, consult the `README.TXT` file in the installation package and the chapter "Installation of R&S CompactTSVP and R&S GTSL" of the "R&S System Manual TSVP".

2.2 Installation

Refer to the "Installation" chapter of the R&S GTSL user manual.

Make sure that the "Enhanced GTSL" checkbox is marked in the "Select Features" dialog of the setup procedure.

2.3 File Structure

The test libraries supplied by ROHDE & SCHWARZ are stored in fixed directories at the time of installation. The following directory structure can be found as subdirectories below the R&S GTSL program files directory which was specified during the installation process:
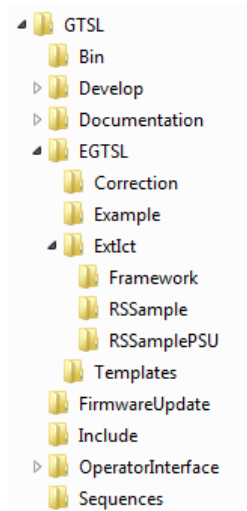
Figure 2-1: File structure program files

Description of installed R&S GTSL program files directories:

Table 2-1: Program files directories

Directory	Contents
GTSL	Generic Test Software Library. The root directory for the R&S GTSL software can have any name.
Bin	Contains the test libraries (DLL, LIB) and the help files (HLP, CHM) belonging to the test libraries.
Develop	The subdirectories of the directory Develop contain generally valid examples for the creation of a high-level test library, a customer-specific selftest library and two sample applications that show how to call R&S GTSL functions and driver functions.
Documentation	Contains the various items of documentation in PDF file format.
EGTSL • Correction	Contains the utility ICTCorrection with which a correction data set can be determined.

EGTSL
•   ExtIct
   –   Framework
   –   RSSample
   –   RSSamplePSU
	The folder …\Framework contains the framework code for the creation of a user-defined ICT extension library. The folder …\RSSample contains an example for the creation of a user-defined ICT extension library with source-code and binaries. The RSSamplePSU project is similar to the RSSample project, but uses the R&S TS-PSU module instead of the R&S TS-PFG module.

EGTSL • Templates	Contains a template for the preparation of a new ICT program on the R&S EGTSL user interface
Firmware Update	This directory contains the firmware update application with which the firmware of all TSVP Test System Versatile Platform plug-in boards can be updated to a newer version when available.

Directory	Contents
Include	Contains the h-files (include files) needed for the development of new test libraries.
Operator Interface	Contains the run time module for the operator interface of TestStand. A TestStand run time licence is required.
Sequences	Contains the test sequence examples created by ROHDE & SCHWARZ.

The application data is stored in the following directory structure below the R&S GTSL application data directory which was specified during the installation process.
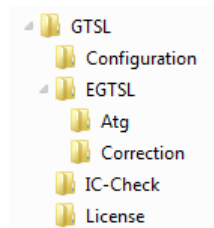
Figure 2-2: File structure application data

Description of installed R&S GTSL application data files directories.

Table 2-2: Application data files directories

Directory	Contents
Configuration	Contains samples of the two configuration files PHYSICAL.INI and APPLICATION.INI.
EGTSL • Atg	Contains the files for the execution of the Automatic Test Generator ATG.
EGTSL • Correction	Contains the correction data determined with the utility ICTCorrection.
IC-Check	Contains example configuration data for the IC-Check application.
License	Contains one or more subdirectories with optional license key files.

3 Functional Description

3.1 General
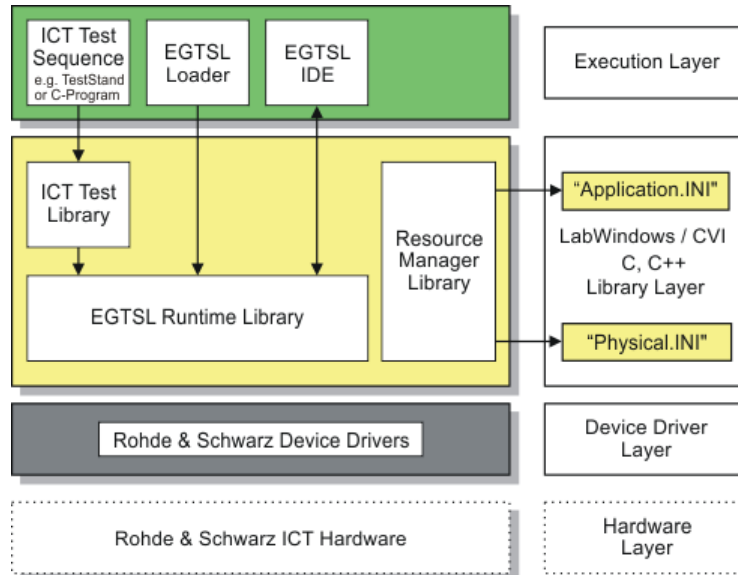
Figure 3-1: R&S EGTSL Layer Model

The Enhanced Generic Test Software Library R&S EGTSL is part of an extension to the Generic Test Software Library R&S GTSL. Using R&S EGTSL it is possible to prepare and perform in-circuit tests. The individual R&S EGTSL software components are arranged in individual layers exactly like R&S GTSL.

The bottom layer (device driver layer) of the R&S EGTSL contains the device drivers necessary for the test hardware used. The test hardware is accessed using these device drivers.

The middle layer (library layer) of the R&S EGTSL contains the ICT test library and the R&S EGTSL. The ICT test library provides the functions necessary for performing the in-circuit test. Using the R&S EGTSL runtime library (runtime engine) the internal R&S EGTSL program processes are run. In this layer further information is passed to the resource manager library via the two files PHYSICAL.INI and APPLICATION.INI. The various device drivers from the lowest level are called from this layer.

The top layer (execution layer) contains the test sequences for performing the in-circuit test. The test sequences call functions from the ICT test library in the middle layer. The function calls include, e.g.

● loading ICT programs,

● running ICT programs,

● debugging ICT programs and

● the generation of reports.

The calls for the individual functions from the ICT test library can, e.g., be made using a sequence editor (TestStand) or a dedicated C program.

The top layer (execution layer) also contains the R&S EGTSL Loader and the R&S EGTSL user interface (R&S EGTSL). The R&S EGTSL user interface (R&S EGTSL) is opened using the R&S EGTSL either by the R&S EGTSL Loader or a function call from the ICT test library.

Special test hardware is required for performing in-circuit tests. This hardware and thus the individual test functions are called using the R&S GTSL/R&S EGTSL-typical functions in the test libraries.

ⓘ For further information on R&S GTSL see "Software Description Generic Test Software Library R&S GTSL"

R&S EGTSL includes the following parts and programs:

- ICT runtime library
- ICT test library
- User interface (IDE)
- Loader
- Automatic Test Generator ATG (utility)
- ICT correction data (utility)

3.2 ICT Test Library

The following section provides a short overview of the test functions available in the ICT test library.

ⓘ The individual test functions and their parameters are described in the online help for the ICT test library. The help files (.HLP) are in the folder ...\GTSL\BIN.

3.2.1 General

ⓘ Starting with GTSL 3.30, no GTSL license is required.

Name of the dynamic link library (DLL):	ICT.DLL
Name of the help file (HLP):	ICT.HLP

License required	R&S TS-LBAS and
	R&S TS-LEGT or R&S TS-LEG2
Supported devices:	R&S TS-PICT ICT Extension Module
	R&S TS-PMB Matrix Module
	R&S TS-PSAM Source and Measurement Module
	R&S TS-PSU Power Supply / Load Module
	R&S TS-PSU12 Power Supply / Load Module 12V

The in-circuit test library offers functions for the in-circuit test using the R&S EGTSL software and the R&S TS-PSAM, R&S TS-PICT, R&S TS-PSU, R&S TS-PSU12 and R&S TS-PMB modules.

The functions allow to

- load, run and debug ICT programs
- load limit files
- generate a report

3.2.2 Entries in PHYSICAL.INI

Section [device->...]

Keyword	Value	Description
Type	String	*Mandatory entry*
		pict = R&S TS-PICT ICT Extension Module
		pmb = R&S TS-PMB Matrix Module
		psam = R&S TS-PSAM Source and Measurement Module
		psu = R&S TS-PSU Power Supply/Load Module
		psu12 = R&S TS-PSU12 Power Supply/Load Module 12V
ResourceDesc	String	*Mandatory entry*
		VISA resource descriptor in the form
		PXI[segment number]::[device number]::[function]::INSTR
		CAN[board]::[controller]::[frame]::[slot]

Keyword	Value	Description
DriverPrefix	String	*Mandatory entry* Prefix for the IVI driver functions, without underscore: R&S TS-PICT : rspict R&S TS-PMB : rspmb R&S TS-PSAM : rspsam R&S TS-PSU: rspsu R&S TS-PSU12: rspsu
DriverDLL	String	*Mandatory entry* File name of the driver DLL R&S TS-PICT : rspict.dll R&S TS-PMB : rspmb.dll R&S TS-PSAM : rspsam.dll R&S TS-PSU : rspsu.dll R&S TS-PSU12 : rspsu.dll
DriverOption	String	*Optional entry* Option string being passed to the device driver during the Driver_Init function. See the online help file for the appropriate device driver.

3.2.3 Entries in APPLICATION.INI

Section [bench->...]

Keyword	Value	Description
ICTDevice1	String	*Mandatory entry* Refers to the device section of the R&S TS-PSAM
ICTDevice2	String	*Optional entry* Refers to the device section of the R&S TS-PICT or R&S TS-PSU / R&S TS-PSU12
ICTDevice3	String	*Optional entry* Refers to the device section of the R&S TS-PICT or R&S TS-PSU / R&S TS-PSU12

Keyword	Value	Description
ICTCorrResistor2W	REAL	*Optional entry* Unit: Ohm Bench correction value, that is taken into account for all 2-wire resistor test measurements (DC-measurements). The correction value is subtracted from the measurement. The result of the subtraction is compared with the limits.
ICTCorrImpedanceRes2W		
	REAL	*Optional entry* Bench correction value, that is taken into account for all 2-wire impedance test resistor measurements (AC-measurements). The correction value is subtracted from the measurement. The result of the subtraction is compared with the limits.
ICTCorrImpedanceCap2W		
	REAL	*Optional entry* Bench correction value, that is taken into account for all 2-wire impedance test capacitance measurements. The correction value is subtracted from the measurement. The result of the subtraction is compared with the limits.
ICTCorrImpedanceInd2W		
	REAL	*Optional entry* Unit: Henry Bench correction value, that is taken into account for all 2-wire impedance test inductance measurements. The correction value is subtracted from the measurement. The result of the subtraction is compared with the limits.
SwitchDevice<i>	String	*Mandatory entry* Refers to a section with switch devices in PHYSICAL.INI. <i> stands for a number from 1,2,3,...,n. The numbers must be assigned in ascending order without gaps. <i> may be omitted in the case it is 1.

Keyword	Value	Description
AppChannelTable	String	*Mandatory entry* Refers to a section with defined channel names in APPLICA-TION.INI.
Simulation	0 / 1	*Mandatory entry* Blocks the simulation of the entered devices (value = 0). Enables simulation of the entered devices (value = 1). Default = 0
Trace	0 / 1	*Optional entry* Blocks the tracing function of the library (value = 0). Enables the tracing function of the library (value = 1). Default = 0
ChannelTableCase Sensitive	0 / 1	*Optional entry* The channel names in the channel table are treated case-sensitive (value = 1) or case-insensitive (value = 0).
Compensation	1 / 0	*Optional entry* Enables the fixture compensation mechanism in the EGTSL/ICT modules. If this option is set to 0, the module will not write any compensation info into the outgoing program file. Default = 0

Section [io_channel->...]

Contains a list of user-specific channel names (or ATG-defined channel names) which are assigned to the physical device names and to the physical device channel names. The defined names apply only to the relevant application. For details about channel name syntax, see "Software Description GTSL", chapter 8.3.4.

Keyword	Value	Description
<user-defined name>	String	Physical channel description in the combination <device name>! <device channel name>

3.2.4 Functions

```
Setup                        ICT_Setup
Library Version              ICT_Lib_Version
EGTSL Runtime Version        ICT_Runtime_Version
Program Control
```

```
    Load Program              ICT_Load_Program
    Run Program               ICT_Run_Program
    Debug Program             ICT_Debug_Program
    Unload Program            ICT_Unload_Program
Report Generation
    Write Report to File      ICT_Write_Report
    Load Detailed Report      ICT_Load_Detailed_Report
    Get Detailed Report Entry ICT_Get_Detailed_Report_Entry
    Get Detailed Report Entry (Extended)
                              ICT_Get_Detailed_Report_Entry_Ex
    Get TestStand Report Entry ICT_Get_TestStand_Report_Entry
    Transfer Report to QUOTIS ICT_Transfer_Quotis_Report
Limit Loader
    Load Limits               ICT_Load_Limits
Error Handling
    Get Error Log             ICT_Get_Error_Log
Cleanup                       ICT_Cleanup
```

3.3 R&S EGTSL Loader

Using the **R&S EGTSL Loader** the R&S EGTSL user interface (*R&S EGTSL IDE*) can be opened directly. A function call from the ICT test library is not necessary.

The operation of the R&S EGTSL Loader is described in section 13.2.

3.4 R&S EGTSL user interface (R&S EGTSL IDE)

The R&S EGTSL user interface (*R&S EGTSL IDE*) for editing and debugging an ICT program is started either using a function from the ICT test library or using the R&S EGTSL Loader.
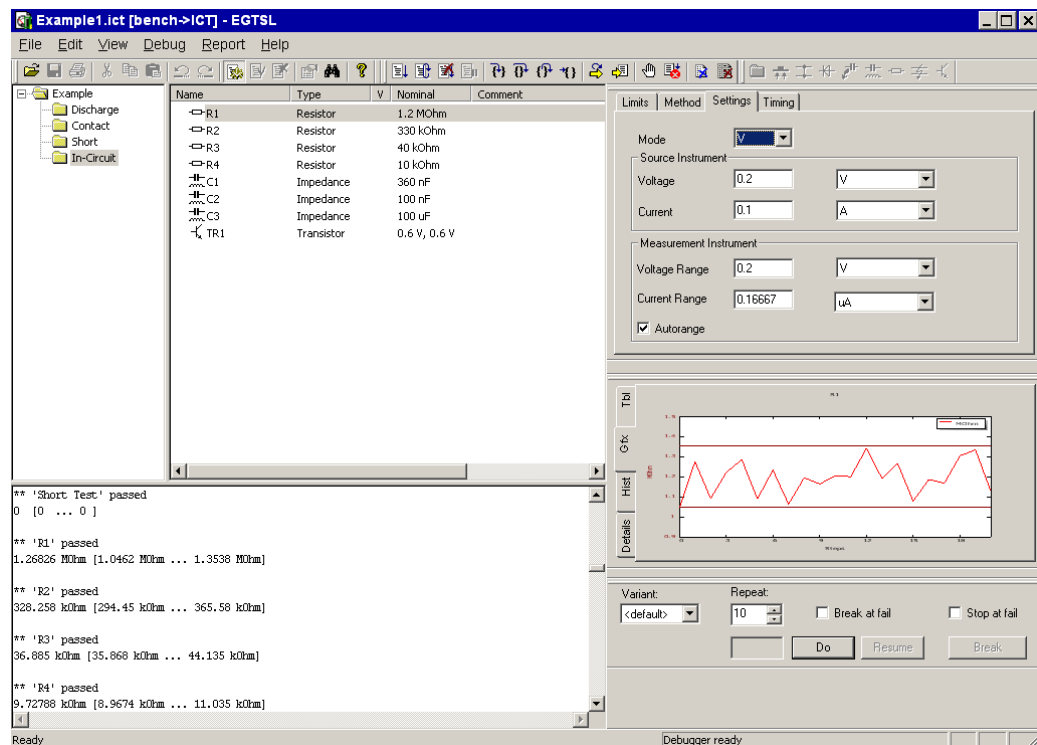
Figure 3-2: R&S EGTSL User interface (R&S EGTSL IDE)

The R&S EGTSL user interface (R&S EGTSL IDE) provides a wide range of functions for preparing, editing and debugging an ICT program.

● Loading ICT programs

● Editing general ICT program settings

● Importing and exporting values for limits for all test steps

● Printing the test report generated

● Inserting and deleting test steps

● Editing the test step settings

● Search functions

● Setting break points

● Debugging the ICT program

– Complete execution of the ICT program

– Execution in single steps

– Execution until an error occurs

– Execution of marked test steps

● Display of the test results (text and graphics)

● Preparation of a test report

ⓘ   The individual functions of the R&S EGTSL user interface (R&S EGTSL IDE) are described in section 5.

3.5 Automatic Test Generator ATG

Using the Automatic Test Generator ATG utility, a circuit description and the description of the test hardware available are used to generate an ICT program that will run in the Enhanced Generic Test Software Library R&S EGTSL. The ICT program generated can be opened in the R&S EGTSL user interface (*R&S EGTSL IDE*) and edited.

ⓘ The individual functions of the Automatic Test Generator ATG are described in section 9.

3.6 ICT Correction Data

The test system is calibrated using the ICTCorrection utility. Resistances and capacitances in the system are determined and saved as correction data. Using this data, the values measured during the in-circuit test are corrected.

ⓘ The individual functions of the ICTCorrection program are described in section 15.

3.7 Hardware for in-circuit test

To be able to perform an in-circuit test using R&S EGTSL the following test hardware must be available:

- Test System Versatile Platform R&S CompactTSVP
- Test System Versatile Platform R&S PowerTSVP (alternative expansion)
- R&S TS-PSAM Source and Measurement Module
- R&S TS-PICT In-Circuit Test Module (alternative expansion for certain test methods)
- R&S TS-PSU Power Supply/Load Module
- R&S TS-PSU12 Power Supply/Load Module 12V (expansion for zener diode and transistor test methods).
- R&S TS-PMB Matrix Module B
- Test adapters developed and built especially for the Unit Under Test (UUT) and the related test program

ⓘ The precise composition and number of items of hardware is always dependent on the specific project and the test methods required.

The individual components of test hardware are described in the related user manuals.

4 Getting Started

4.1 General

This section describes the development process for an in-circuit test in a step-by-step manner based on a simple example. The individual steps can be followed using these instructions so that you can rapidly become familiar with the Enhanced Generic Test Software Library R&S EGTSL.

ⓘ The example circuit, the related BDL file and the files generated by the Automatic Test Generator ATG are listed in appendix A. The related files are saved in `...\GTSL\EGTSL\Example`.

4.2 Development Process for an In-circuit Test

Based on the (existing) circuit documents for a unit under test (for the in-circuit test the unit is generally a circuit board) an ICT program is to be prepared that can test whether the circuit board is correctly populated. The mechanical-electrical adaptation of the unit under test to the test system, i.e. building a bed of nails adapter is also included.

The Enhanced Generic Test Software Library R&S EGTSL provides a series of utilities that enable these tasks to be tackled largely automatically. Based on an example circuit, the features provided by the software will be demonstrated.

To pass from the circuit documentation to the finished ICT program, the following steps are necessary:

- Preparation of the circuit documentation
- Entry of the circuit description
- Preparation of the test program by the Automatic Test Generator ATG
- Preparation of the adapter
- Commissioning and debugging the test program

4.3 Preparation of the Circuit Documentation

In the first step the test points, i.e. the pins on the bed of nails adapter, are entered on the circuit diagram. As during the in-circuit test, every component is measured separately, it is important that the test system can make contact to every pin on every component. For this purpose, a test nail must be provided at every node of the circuit.

The names of the test points, also called nodes, appear later in the test program. They must therefore be clear, i.e. either refer to the function in the circuit (like INPUT, OUTPUT) or to a component connected to the node (like TR1.B for the base of transistor

TR1). Special rules apply for ground and supply voltage. The names of such test points must always start with GND or VCC respectively, so that the automatic test generator can detect them as ground and supply voltage and treat them appropriately.

4.3.1 Example Circuit
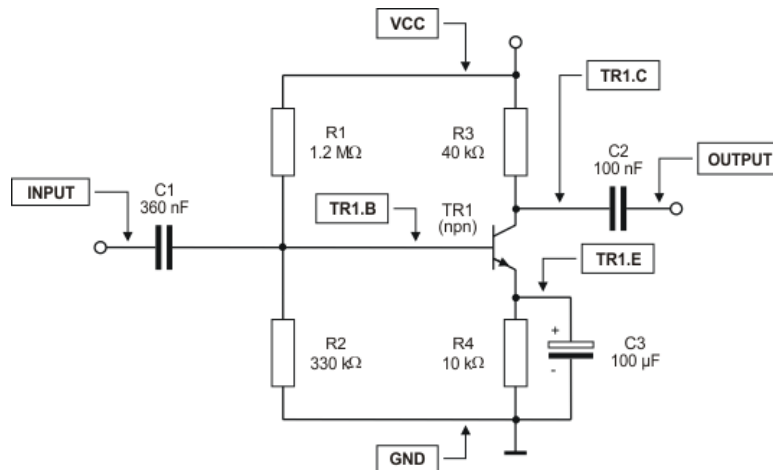
Figure 4-1: Circuit example

Figure 4-1 shows the circuit for a low frequency amplifier stage. The names of the test points are already entered.

4.3.2 Preparation of the Circuit Description

For further processing in R&S EGTSL the circuit description for the circuit to be tested must be available as a BDL file (see also section 8.2). There are now various ways you can prepare the necessary BDL file.

1. Prepare the BDL file (circuit description) manually in a text editor based on the circuit diagram.

2. Transfer the circuit description from a CAD system. Modern CAD systems for the preparation of electronic circuits provide a feature for exporting the circuit description. Using commercially available conversion software, the exported circuit description can be converted to the BDL format.

The manual preparation of the circuit description is of course tedious for larger circuits. Here it is best, if possible, to transfer the data from the CAD system. For the small example with only eight components, it is however easy to enter the BDL data by hand.

The BDL file is saved in `...\GTSL\EGTSL\Example\Example1.BDL`.

4.3.2.1   Entry of the Resistors

BDL example for resistor R1

```
RESISTOR

NAME 'R1'
     PART_ID '1234.5678.90'
     VALUE 1.2 MOHM
     TOL+ 10%  TOL- 10%
     I_LIM 100.0 MA
     PIN_1 'VCC'
     PIN_2 'TR1.B'
     ERR_MSG 'Problem with R1'
```

The following information on a resistor must be entered in the text editor:

NAME	Component name
PART_ID	Optional component identifier for the component
VALUE	Nominal value and unit for the component
TOL+	Positive component tolerance in % (default value 10 %)
TOL-	Negative component tolerance in % (default value 10 %)
I_LIMIT	Maximum measuring current for determining the value with unit (default value 100 mA)
PIN_1	Test point 1 for the component
PIN_2	Test point 2 for the component
ERR_MSG	Optional error text The text entered here is displayed in the report if the component has been detected as faulty.

For the example circuit, all resistors must be entered in the same manner:

Name	Value	Pin 1	Pin 2
R1	1.2 MOHM	VCC	TR1.B
R2	330 KOHM	TR1.B	GND
R3	40 KOHM	VCC	TR1.C
R4	10 KOHM	TR1.E	GND

4.3.2.2   Entry of Capacitors

BDL example for capacitor C1

```
CAPACITOR
```

```
NAME 'C1'
     PART_ID '0987.6543.21'
     VALUE 360 NF
     TOL+ 10%  TOL- 10%
     PIN_1 'INPUT'
     PIN_2 'TR1.B'
     ERR_MSG 'Problem with C1'
```

The following information on a capacitor must be entered in the text editor:

NAME	Component name
PART_ID	Optional component identifier for the component
VALUE	Nominal value and unit for the component
TOL+	Positive component tolerance in % (default value 10 %)
TOL-	Negative component tolerance in % (default value 10 %)
PIN_1	Test point 1 for the component
PIN_2	Test point 2 for the component
ERR_MSG	Optional error text The text entered here is displayed in the report if the component has been detected as faulty.

For the example circuit, all capacitors must be entered in the same manner:

Name	Value	Pin 1	Pin 2
C1	360 NF	INPUT	TR1.B
C2	100 NF	TR1.C	OUTPUT

When entering electrolytic capacitors, attention must be paid to two special aspects: the unit microfarad is identified using a "U". The positive pole is termed pin A (= anode), the minus pole as pin C (=cathode).

BDL example for capacitor C3 (electrolytic)

```
POL_CAP

NAME 'C3'
     PART_ID 'C300.4711'
     VALUE 100 UF
     TOL+ 10%  TOL- 10%
     PIN_A 'TR1.E'
     PIN_C 'GND'
     ERR_MSG 'Problem with C3'
```

Name	Value	Pin A	Pin C
C3	100 UF	TR1.E	GND

4.3.2.3 Entry of the Transistor

BDL example for transistor T1

```
TRANSISTOR

NAME 'TR1'
      PART_ID 'T1000'
      NPN
      UBE 0.6 V
      TOL+ 30%   TOL- 30%
      I_LIM 100.0 MA
      PIN_E 'TR1.E'
      PIN_B 'TR1.B'
      PIN_C 'TR1.C'
      ERR_MSG 'Problem with T1'
```

The following information on a transistor must be edited in a text editor:

NAME	Component name
PART_ID	Optional component identifier for the component
NPN, PNP	Type of transistor
UBE	Base-emitter voltage of the transistor in the forward bias region
TOL+	Positive component tolerance in %
TOL-	Negative component tolerance in %
I_LIMIT	Maximum measuring current for determining the value with unit (default value 100 mA)
PIN_E	Test point for the transistor's emitter
PIN_B	Test point for the transistor's base
PIN_C	Test point for the transistor's collector
ERR_MSG	Optional error text The text entered here is displayed in the report if the component has been detected as faulty.

Name	Type	Pin E	Pin B	Pin C
TR1	N	TR1.E	TR1.B	TR1.C

4.4   Preparing the ICT program

Once the circuit description is available as a BDL file, an ICT program can be prepared with the aid of the Automatic Test Generator ATG.

The Automatic Test Generator ATG is described in section 9.2.

Automatic Test Generator	Start the Automatic Test Generator ATG using **Start -> Programs -> GTSL -> Automatic Test Generator**.

The following files must be entered in the ATG:

- `Example1.BDL`
  path and file name for the BDL file (circuit description)
- Example_Physical.INI
  path and file name for the Physical Layer Configuration File (hardware description of the test system)

Following the automatic generation process, the ATG prepares the following files:

- `Example1_Report.TXT`
  report with the data on the generation process
  Any errors that have occurred during the generation process are indicated in the report using error messages and warning messages (see section 9.3.2.1). Only when there are no error messages or warning messages is the ICT program prepared. In addition, the alternative suggestions (proposals) made by the ATG are to be noted (see section 9.3.2.2).
- `Example1.ICT`
  ICT program for execution in R&S EGTSL
  The layout of the automatically generated ICT program is defined by a structure anchored in the ATG. In principle, the fully generated test program has the following layout:
  - Capacitor discharging (Discharge)
  - Contact test (Contact)
  - Continuity and short-circuit test (Short)
  - Group of analog tests (In-Circuit)
- `Example1_Application.INI`
  Application Layer Configuration File for the execution of the ICT program generated in R&S EGTSL
  In the Application Layer Configuration File, the specific information for the usage of the hardware is compiled by the ATG for the ICT program generated. On the execution of the ICT program, the name of the Application Layer Configuration File generated must be given.
- `Example1.ICC`

ICC program for execution with IC-Check program. See Chapter 9.3.5, "ICC Program", on page 194 and "Software Description IC-Check" for details.

4.5 Adapter Manufacture

In the Application Layer Configuration File generated by the Automatic Test Generator ATG, an I/O channel on the R&S TS-PMB Matrix Module B is allocated to each test point (node) of the circuit.

Example:

```
[io_channel->ICT]
GND         = PMB1!P1
INPUT       = PMB1!P2
OUTPUT      = PMB1!P3
TR1.B       = PMB1!P4
TR1.C       = PMB1!P5
TR1.E       = PMB1!P6
VCC         = PMB1!P7


[io_wiring->ICT]
GND         = F1 S15 X10A1
INPUT       = F1 S15 X10A2
OUTPUT      = F1 S15 X10A3
TR1.B       = F1 S15 X10A4
TR1.C       = F1 S15 X10A5
TR1.E       = F1 S15 X10A6
VCC         = F1 S15 X10A7
```

This information can be used for the manufacture of the adapter.

4.6 Commissioning and Debugging

The newly prepared ICT program is now, together with the finished adapter, ready to use. However, correct operation should first be checked on a few circuit boards. It may be necessary to slightly change test value limits or adjust settings.

The user interface for R&S EGTSL (*R&S EGTSL IDE*) for debugging (and executing) ICT programs can be started in two ways:

1. Start the user interface from a test sequence using a function call from the ICT test library (see section 13.1)

2. Start the user interface using the Loader (see section 13.2)
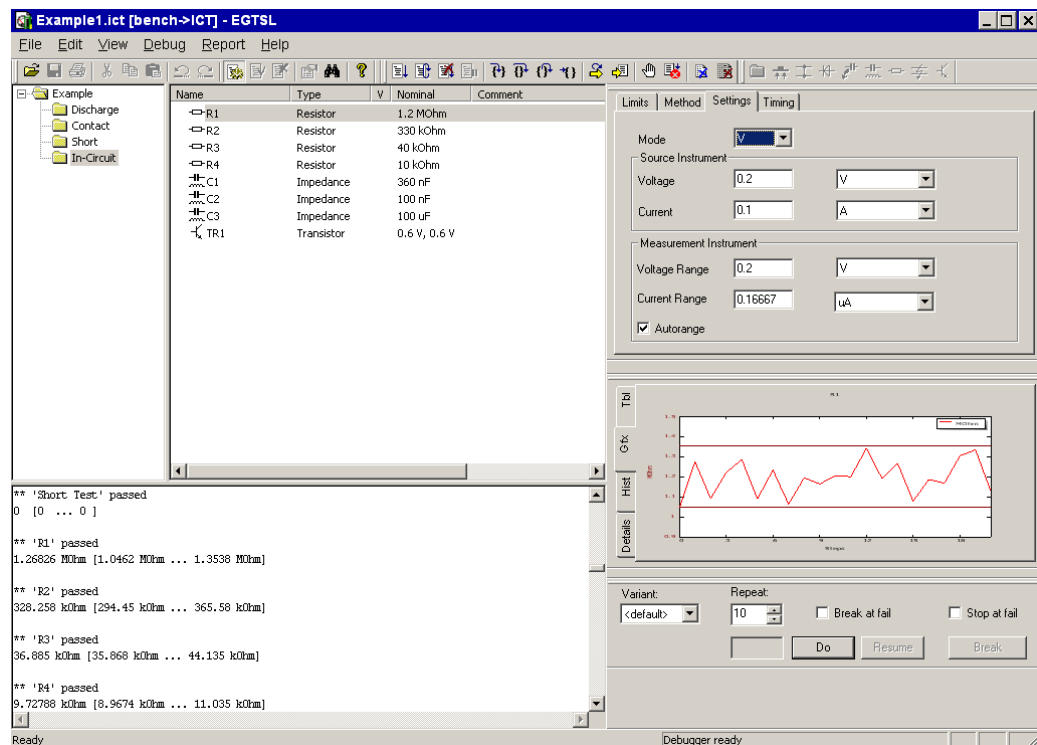
Figure 4-2: R&S EGTSL User Interface (R&S EGTSL IDE)

When starting the *R&S EGTSL IDE* from a test sequence, the ICT program is automatically opened with the corresponding bench. When starting the *R&S EGTSL IDE* using the R&S EGTSL Loader, the ICT program must be opened manually using the menu command **File -> Open**. When opening the ICT file, the corresponding bench for executing the ICT program must be entered.

Modifications can be made to the ICT program in the various windows of the user interface. When the debugger is activated, various types of program execution are possible:

- Complete execution of the ICT program
- Execution in single steps
- Execution until an error occurs
- Execution of marked test steps

The various windows, menu commands and buttons on the R&S EGTSL user interface (R&S EGTSL IDE) are described in section 5. The detailed procedure for debugging ICT programs is described in section 15.

As the example circuit will probable not be built and an adapter has not been produced, the ICT program can only be tested and debugged in the simulation mode. For this purpose it is necessary to activate the simulation in the Application Layer Configuration File in the section [bench->...].

```
[bench->ICT]
Description     = ICT bench (Simulation)
Simulation      = 1
Trace           = 0
ICTDevice1      = device->psam
ICTDevice2      = device->pict
SwitchDevice1   = device->pmb1
AppChannelTable = io_channel->ICT
```

When the ICT program is opened with the corresponding bench, all functions in the ICT program and the R&S EGTSL user interface can be executed in the simulation mode.

5 User Interface (R&S EGTSL IDE)

5.1 General

The EGTSL debugging environment has the familiar look of modern Windows applications.

With this graphical user interface, you can quickly master the full spectrum of functionality. There is no need to learn a special programming language or work with a predefined screen layout. Test engineers can create their own desktop and display all required information on one screen. They can scale, move, and display the windows as needed.

Within the EGTSL debugging environment, you can do the following:

● Create, delete or move test steps by a mouse click or drag and drop
● Change settings of test steps, e.g. limits, test method, stimuli and measurement
● Define specific timing models
● Use functionalities such as setting breakpoints, step into or over certain test steps
● Define and handle different variants
● Export and import limits
● Use statistical tools such as histograms to verify stable results
● Display a detailed test report.

🛈 To operate the Enhanced Generic Test Software Library R&S EGTSL, basic skills in the usage of the Microsoft Windows operating systems are required.

5.2 Menu Structure

Figure 5-1: Menu structure

5.3 Main Screen
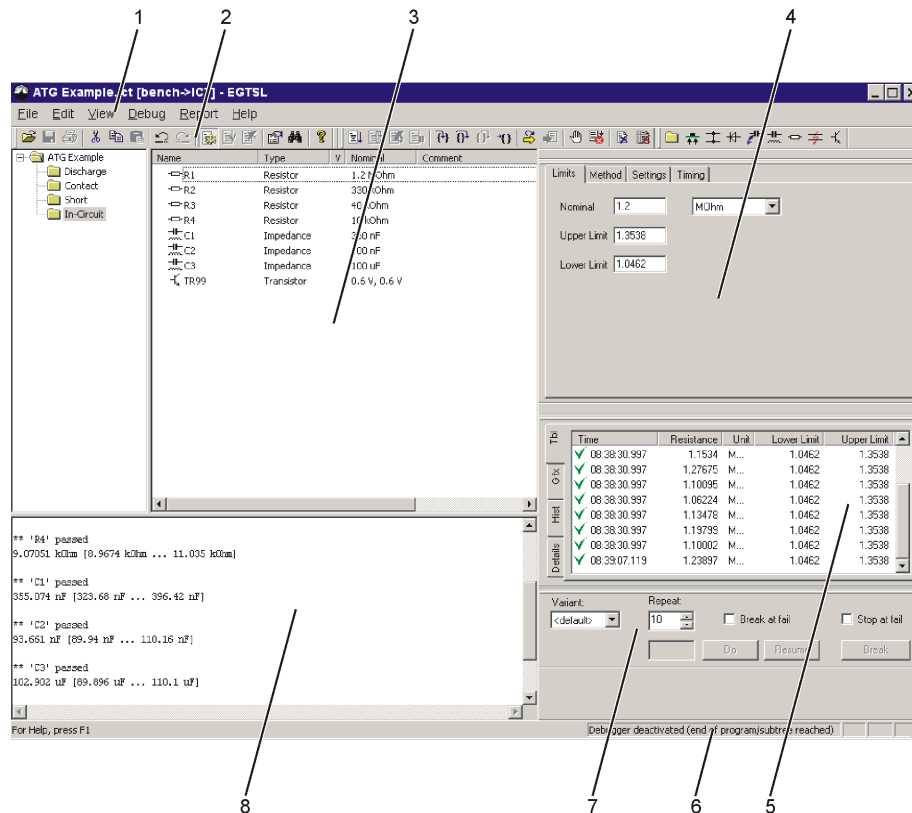
Figure 5-2: Main screen R&S EGTSL

1 = Menu bar (see Chapter 5.6, "Menu bar Functions", on page 110)
2 = Toolbars (see Chapter 5.7, "Toolbar Functions", on page 128)
3 = Program subwindow (seeChapter 5.4.2, "Program Subwindow", on page 40)
4 = Test Properties subwindow (see Chapter 5.4.4, "Test Properties Subwindow", on page 43 and Chapter 5.5, "Test Steps", on page 47)
5 = Results subwindow (see Chapter 5.4.5, "Results Subwindow", on page 43)
6 = Status bar
7 = Debug subwindow (seeChapter 5.4.6, "Debug Subwindow", on page 46)
8 = Report subwindow (see Chapter 5.4.3, "Report Subwindow", on page 42)

5.4 Subwindow

5.4.1 Positioning of the Subwindows

See Figure 5-2.

The Program and Report subwindows have a fixed position on the left of the R&S EGTSL screen. The vertical division between the two windows can be varied using the mouse.

The Test Properties, Result and Debug subwindows are displayed on the right of the R&S EGTSL screen in a fixed position with a fixed size. The size of the Program and Report subwindows is automatically adjusted when the Test Properties, Result and Debug subwindows are displayed. The position of the Test Properties, Result and Debug subwindows can be changed using the mouse. On the right of the R&S EGTSL screen the Test Properties, Result and Debug subwindows dock in a fixed position when moved. When the **Ctrl** key is pressed, the Test Properties, Result and Debug subwindows can be moved to any position using the mouse (no docking).

The size of the Result subwindow can be changed using the mouse (horizontal and vertical). By double-clicking the Result subwindow, you can change between the freely selectable position (and size) and the fixed position on the right of the R&S EGTSL screen.

5.4.2 Program Subwindow
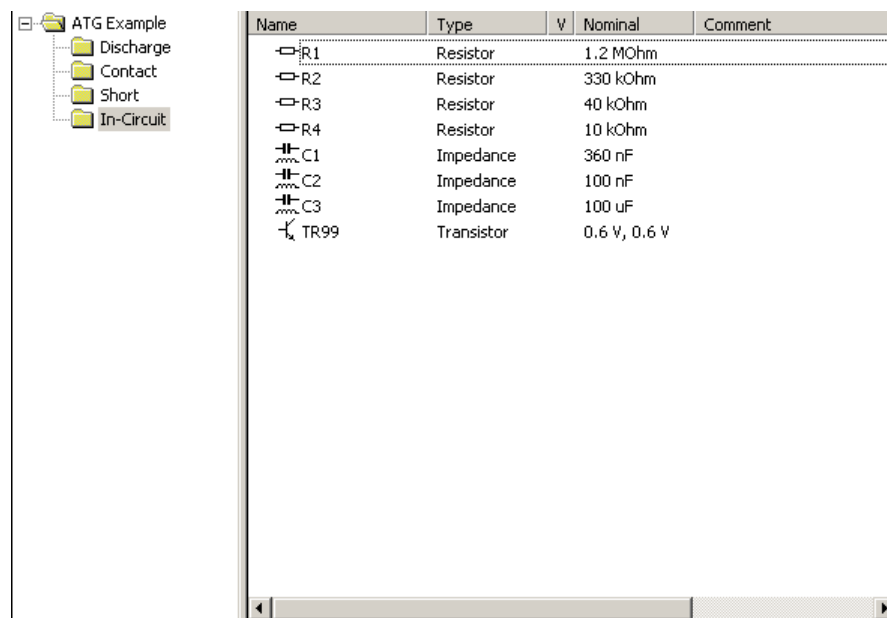
Figure 5-3: Program subwindow

The ICT program is displayed on the left of the Program subwindow together with its individual program groups in a tree structure. The individual test steps in the ICT program are displayed on the right of the Program subwindow.

The horizontal distribution between the two windows can be varied using the mouse.

The navigation within the two windows and the editing of test steps is performed using the familiar Windows mouse and key commands, e.g.:

Left mouse button pressed	Move and mark test steps and groups
Ctrl + left mouse button	Mark several test steps

Shift + left mouse button	Mark a range of test steps
Right mouse button	Open the context menu

5.4.2.1　Context Menu

When you click the right mouse button in the Program window, a context menu is opened.
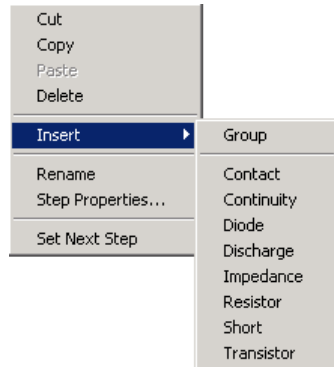
Figure 5-4: Program context menu

"Cut"	Deletes the marked test step or the marked program group and copies it to the Clipboard.
"Copy"	Copies the marked test step or the marked program group to the Clipboard.
"Paste"	Inserts the contents of the Clipboard (test step, program group, text content) at the current cursor position.
"Delete"	Deletes the marked test step or the marked program group.
"Insert"	Opens the list box for inserting test steps or program groups. On this topic, see also Chapter 5.5, "Test Steps", on page 47.
"Rename"	Enables the marked test step name or program group name to be edited.
"Step Properties"	Opens the dialog box for the step properties for the selected test step or the selected program group. On this topic, see also Chapter 5.6.2.3, "Menu Command <Step Properties>", on page 121.
"Set Next Step"	Marks the cursor position as the test step that is to be executed next. A yellow arrow is displayed on the left beside the selected test step.

5.4.3 Report Subwindow

```
** 'R1' passed
1.23897 MOhm [1.0462 MOhm ... 1.3538 MOhm]

** 'R2' passed
302.445 kOhm [294.45 kOhm ... 365.58 kOhm]

** 'R3' passed
41.191 kOhm [35.868 kOhm ... 44.135 kOhm]

** 'R4' passed
9.07051 kOhm [8.9674 kOhm ... 11.035 kOhm]

** 'C1' passed
355.074 nF [323.68 nF ... 396.42 nF]

** 'C2' passed
93.661 nF [89.94 nF ... 110.16 nF]

** 'C3' passed
102.902 uF [89.896 uF ... 110.1 uF]
```

Figure 5-5: Report subwindow

All messages from the ICT are displayed in the Report window. These messages include:

- Information on report preparation
- The test result for each test step with the measured values determined.
- Errors during the compilation of the test steps or the ICT program.

The vertical size of the Report window can be changed by dragging using the mouse.

Additional information on the report format is described in Chapter 14, "Report Format", on page 242.

5.4.3.1 Context Menu

When you click the Report window using the right mouse button, a context menu is opened.

```
  Copy
  Select All
  Clear
✔ Report Enabled
  Errors Only
```
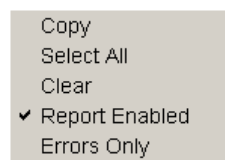
Figure 5-6: Report context menu

"Copy"	Copies all marked entries in the Report window to the Clipboard. Using the familiar Windows keyboard and mouse commands, you can mark the entries in the Report window.
"Select All"	Marks all entries in the Report window.
"Clear"	Deletes all entries in the Report window.
"Report Enable"	With this function activated, a report is displayed in the Report window.
"Errors Only"	With this function activated, only erroneous measurements and error messages are displayed in the Report window.

5.4.4　Test Properties Subwindow

In the Test Properties subwindow, the test step-specific settings are made. The various windows are displayed for every type of test step.

The functions in the individual windows are described in Chapter 5.5, "Test Steps", on page 47 with the individual test steps.

5.4.5　Results Subwindow

The measured results from the individual test steps of the ICT program are displayed in the Result windows in various ways.

5.4.5.1　Context Menu

When you click the right mouse button in the Tbl, Gfx and Hist result windows, the following context menu is opened.
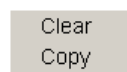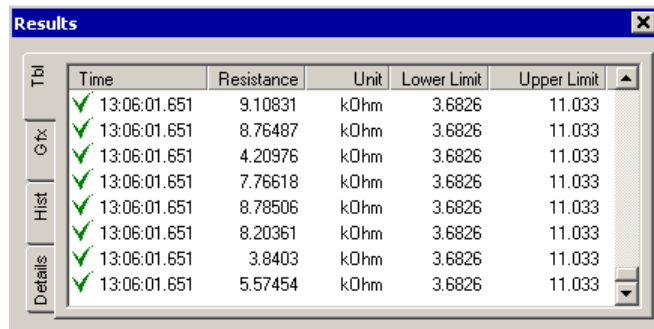
Figure 5-7: Results context menu

"Clear"	Deletes all entries in the Result windows (Tbl, Gfx, Hist) for all test steps.
"Copy"	**Tbl** Copies all test entries for the test step marked to the Clipboard. **Gfx** and **Hist** Copies the graphic for the test step marked to the Clipboard.

Using the Clipboard, you can copy test and graphics to other programs. There you can, e.g., edit, save or print the information copied.

5.4.5.2 Results Tbl

Figure 5-8: Results Table

In the **Results/Tbl** window, the test results for the test step marked are displayed in a table. The information displayed varies depending on the type of test step selected (measured values, values for limits, etc.).

5.4.5.3 Results Gfx
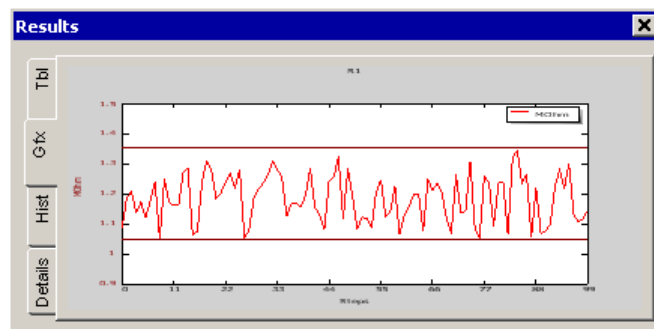
Figure 5-9: Results Graphic

In the **Results/Gfx** window, the test results for the test step marked are displayed graphically. The information displayed varies depending on the type of test step selected. The measured results for the individual measurements and the stipulated values for the limits (horizontal lines) are displayed.

If a test step provides several test results (e.g. diode, transistor), the first test result is displayed in red. The second test result is displayed in blue.

5.4.5.4 Results Hist
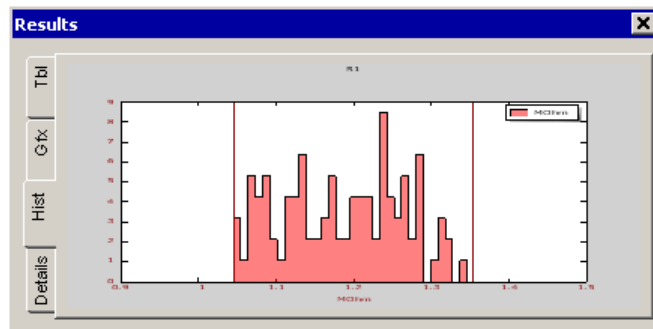
Figure 5-10: Results History

In the **Results/Hist** window, the distribution of the measured results for the test step marked is displayed graphically. The information displayed varies depending on the type of test step selected. In addition, the stipulated values for the limits are displayed as vertical lines.

If a test step provides several test results (e.g. diode, transistor), the first test result is displayed in red. The second test result is displayed in blue.

5.4.5.5 Results Details
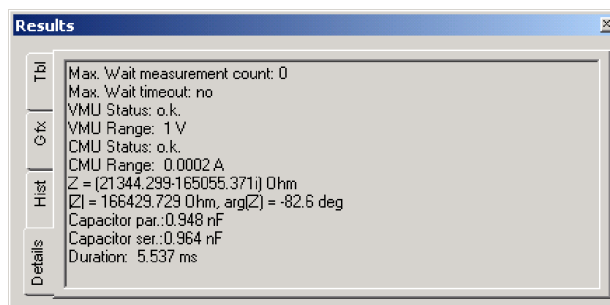
Figure 5-11: Results Details

In the **Results/Details** window, detailed information on the test step marked is displayed. The information displayed varies depending on the type of test step selected.

When you click the right mouse button in the Result window, the following context menu is opened.
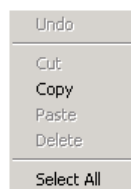
Figure 5-12: Results context menu

"Undo"	This function is not available because the **Results/Details** window is "read only".
"Cut"	This function is not available because the **Results/Details** window is "read only".
"Copy"	Copies the text entry marked to the Clipboard.
"Paste"	This function is not available because the **Results/Details** window is "read only".
"Delete"	This function is not available because the **Results/Details** window is "read only".
"Select All"	Marks all entries in the Results window.

5.4.6　Debug Subwindow
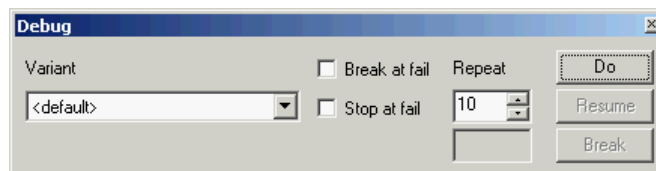
Figure 5-13: Debug

In the **"Debug"** subwindow, the settings for executing (individual) test steps are made. The marked test steps are executed. Only contiguous blocks of test steps can be executed. No test steps can be skipped.

To execute the test steps, the debugger must be started (a test step must be selected using the yellow arrow). The message **Debugger ready** is given in the status bar. The debug mode is activated with the following functions:
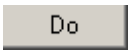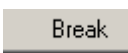
- **Step Into** *[F11]*
- **Step Over** *[F10]*
- **Step Out** *[Shift+F11]*
- **Run to Cursor** *[Ctrl+F11]*
- **Set Next Step** *[Ctrl+Shift+F10]*

Further information on debugging ICT programs is given in Chapter 12, "Debugger", on page 225.

"Variant":	In the **"Variant"** list box, the test steps marked with a variant are enabled for execution. With the selection <default> test steps with a variant marker are not executed. Test steps without a variant marker are always executed.

The variants entered in the Variant list box apply not only to the execution of test steps using the Debug subwindow (see Figure 5-13) but also for all other debug functions (see Chapter 5.6.4, "Main Menu Command <Debug>", on page 125).

" Break at fail"	With this function activated, the execution of the ICT program (test steps marked) is interrupted when the test step result was "Fail". The program stops at the erroneous test step. The **"Break at fail"** function applies only to the **"Repeat"** function.
"Stop at fail"	With this function activated, the ICT program execution is stopped when the test step result was "Fail". The program stops at the erroneous test step. The **"Stop at fail"** function applies to all debug commands with the exception of the **"Repeat"** function.
Repeat:	In the **"Repeat"** field, you can define how often the marked test steps are to be executed.
Do	Starts the execution of the test steps marked with the number of repetitions given
Resume	Continues the execution of the test steps marked on an interruption using the **"Break"** button.
Break	Interrupts the execution of the test steps marked.

5.5 Test Steps

5.5.1 Contact

ⓘ The contact test is described in Chapter 10.3, "Contact Test", on page 197.

5.5.1.1 General
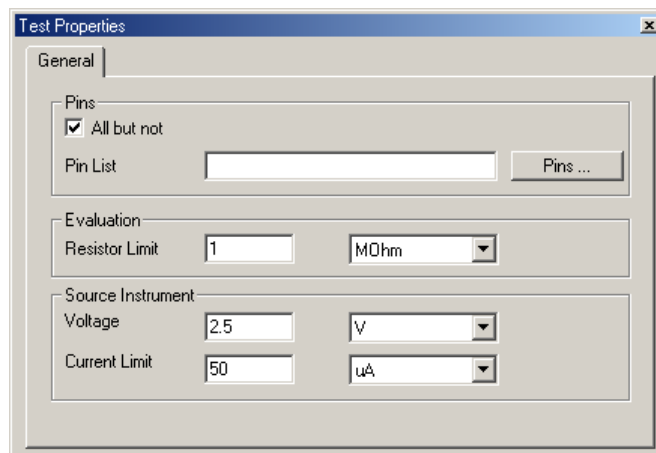
Figure 5-14: Test Step Contact, Test Properties General

Pins	
"All but not"	With this check box selected, all available measuring pins are checked for contact. The list of available measuring pins is read from the "Available Pins" (see Chapter 5.5.12.2, "Editing Pin Lists", on page 105). The measuring pins given in the **"Pin List"** are not checked for contact.
"Pin List"	Check box **"All but not"** cleared: all the measuring pins entered in the Pin List are checked for contact. Check box **"All but not"** selected: all pins except for the measuring pins entered in the Pin List are checked for contact.
Pins ...	Opens the **"Pins"** dialog box for inserting / removing pins on the Pin List. For more information, see Chapter 5.5.12.2, "Editing Pin Lists", on page 105.
Evaluation	
"Resistor Limit"	The maximum resistance that is still evaluated as a contact is entered in this field. Value range: 1 kΩ ... 1 MΩ Default: 1 MΩ
Source Instrument	
"Voltage"	The voltage of the voltage source is entered in this field. Value range: 1.0 V ... 5.0 V Default: 2.5 V
"Current Limit"	The current limit of the voltage source is entered in this field. Value range: 1.0 µA ... 5.0 mA Default: 50 µA

5.5.1.2 Results Details

For the Contact test step, the following information is displayed in the **Results/Details** window after the execution of test step:
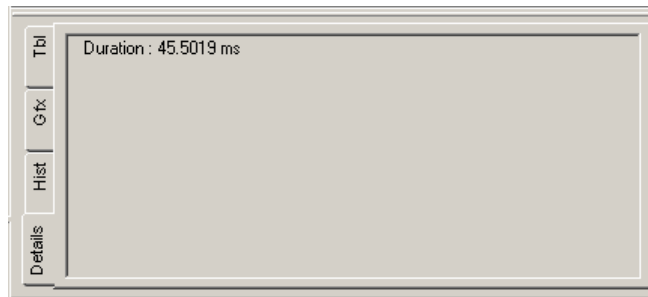
Figure 5-15: Test Step Contact, Results Details

5.5.2 Continuity

ⓘ The continuity test is described in Chapter 10.4, "Continuity Test", on page 198.

5.5.2.1 General
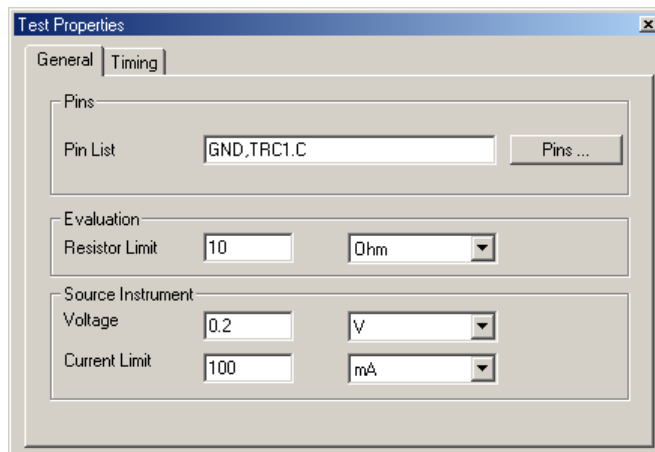
Figure 5-16: Test Step Continuity, Test Properties General

Pins	
"Pin List"	All measuring pins entered in the **"Pin List"** are checked for continuity.
Pins ...	Opens the **"Pins"** dialog box for inserting / removing pins on the Pin List. For more information, see Chapter 5.5.12.2, "Editing Pin Lists", on page 105.
Evaluation	

"Resistor Limit"	The maximum resistance that is still evaluated as continuity is entered in this field. Value range: 1 Ω ... 1 kΩ Default: 10 Ω
Source Instrument	
"Voltage"	The voltage of the voltage source is entered in this field. Value range: 0.1 V ... 0.5 V Default: 0.2 V
"Current Limit"	The current limit of the voltage source is entered in this field. Value range: 1 µA ... 100 mA Default: 100 mA

5.5.2.2 Timing
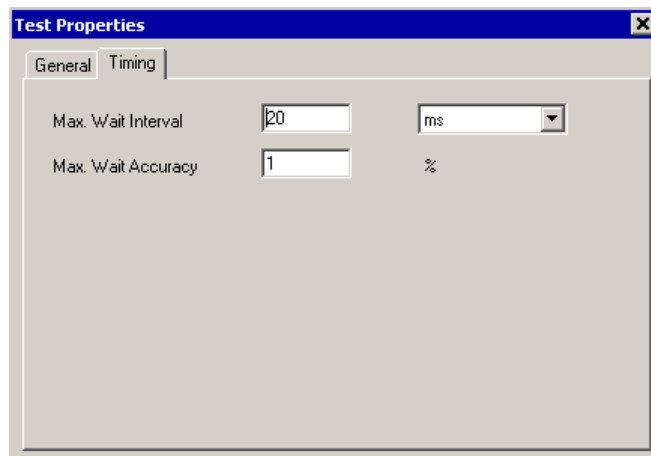
Figure 5-17: Test Step Continuity, Test Properties Timing

ⓘ The individual options for the timing of the measurement operation are described in more detail in

"Max. Wait Interval"	The maximum duration of the test interval for each individual pin is entered in this field (maximum 20 samples). Value range: 0 ms ... 10 s Resolution: 1 ms (at a Max. Wait Interval of 1 ms a sample is taken every 50 µs) Default: 20 ms
"Max. Wait Accuracy"	In this field, the maximum percentage difference between two measured values in succession (samples) is entered for test steps in which the measured value is applied (measurement with "Max. Wait Interval"). Value range: 0.0 % ... 10.0 % Default: 1.0 %

5.5.2.3 Results Details

For the Continuity test step, the following information is displayed in the **Results/Details** window after the execution of test step:
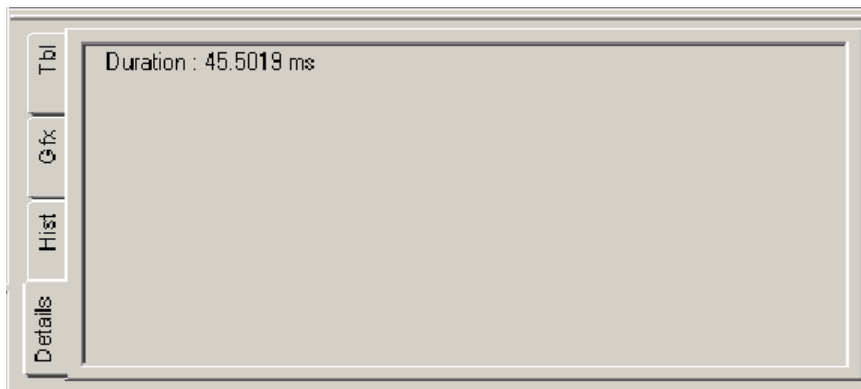
Figure 5-18: Test Step Continuity, Results Details

"Duration"	Duration of the last Continuity test step executed

5.5.3 Diode

The diode test is described in Chapter 10.5, "Diode Test", on page 199.

5.5.3.1 Limits

Figure 5-19: Test Step Diode, Test Properties Limits

"Enable"	During the diode test, two measurements are performed optionally:
	• **"Voltage Measurement"**
	  Measurement of the forward bias voltage (knee voltage)
	• **"Current Measurement"**
	  Measurement of the reverse bias current
	
	The related measurement is performed when the check box is selected. At least one measurement must be activated.

Voltage Measurement	
"Nominal"	The nominal value for the forward bias voltage (knee voltage) to be measured is entered in this field.
"Upper Limit" **"Lower Limit"**	The upper and lower limits of the measured forward bias voltage (knee voltage) can be entered in these fields. The fields on the left contain the absolute value in the same unit as the nominal value. The fields on the right contain the deviation from the nominal value as a percentage. Inactive input fields (which appear in gray) are recalculated and displayed with the command **"Edit… Apply"**.
"Absolute / Relative"	This setting determines whether limits will be entered as absolute values or relative values.
Current Measurement	
"Nominal"	The nominal value for the reverse bias current to be measured is entered in this field.

"Upper Limit"	The upper and lower limits of the measured reverse bias current can be entered in these fields. The fields on the left contain the absolute value in the same unit as the nominal value. The fields on the right contain the deviation from the nominal value as a percentage.
"Lower Limit"	
	Inactive input fields (which appear in gray) are recalculated and displayed with the command **"Edit… Apply"**.
"Absolute / Relative"	This setting determines whether limits will be entered as absolute values or relative values.

5.5.3.2 Settings
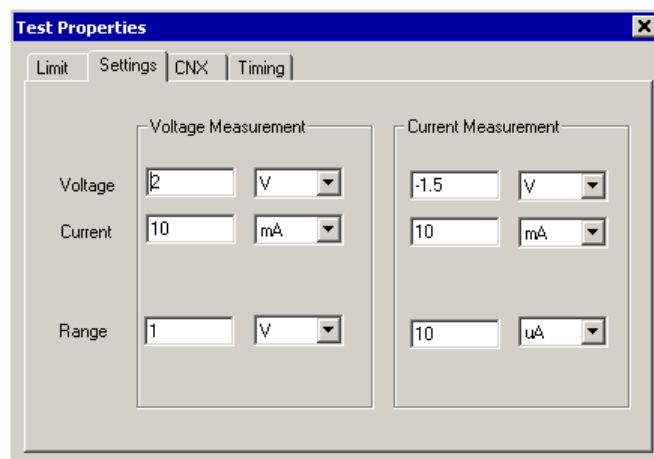
Figure 5-20: Test Step Diode, Test Properties Settings

Voltage Measurement	
"Voltage"	The voltage of the voltage source is entered in this field.
	Value range:
	-5.0 V ... 5.0 V
	Default:
	2.0 V

"Current"	The current limit of the voltage source is entered in this field.
	Value range:
	3 uA ... 100 mA
	Default:
	10 mA
"Range"	The measuring range of the voltmeter is entered in this field.
	Value range:
	10 mV ... 5.0 V
	Default:
	1.0 V

Current Measurement	
"Voltage"	The voltage of the voltage source is entered in this field.
	Value range:
	-5.0 V ... 5.0 V
	Default:
	-1.5 V
"Current"	The current limit of the voltage source is entered in this field.
	Value range:
	1.0 µA ... 100.0 mA
	Default:
	10 mA
"Range"	The measuring range of the current meter is entered in this field.
	Value range:
	0.0 µA ... 200 mA
	Default:
	10.0 µA

5.5.3.3 CNX
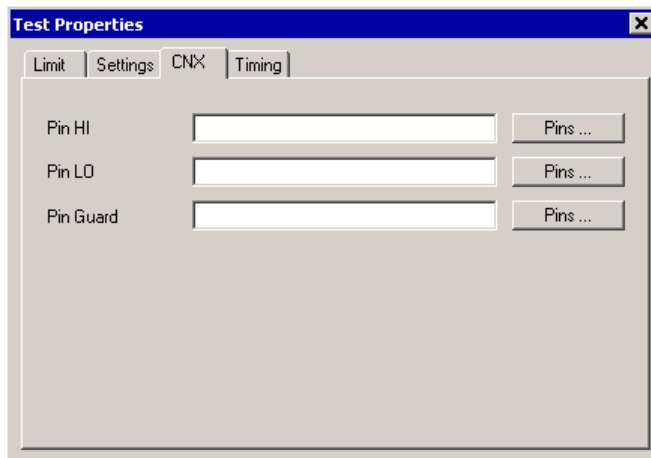
Figure 5-21: Test Step Diode, Test Properties CNX

"Pin HI"	The **Pin HI** (diode anode) is entered in this field. Only one measuring pin can be entered.
"Pin LO"	The **Pin LO** (diode cathode) is entered in this field. Only one measuring pin can be entered.
"Pin Guard"	The **Pin Guard** is entered in this field. Several measuring pins can be entered. The individual entries (measuring pins) are separated with commas.
Pins ...	Opens the **Pins** dialog box for inserting / removing pins on the Pin List. For more information, see Chapter 5.5.12.2, "Editing Pin Lists", on page 105.

5.5.3.4 Timing
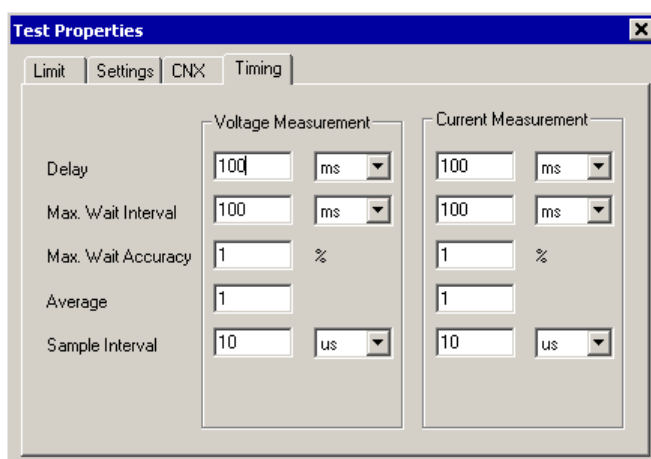
Figure 5-22: Test Step Diode, Test Properties Timing

ⓘ The individual options for the timing of the measuring process are described in more detail in Chapter 5.5.12.1, "Timing", on page 103. The timing settings can be made separately for the *Voltage Measurement* and the *Current Measurement*.

"Delay"	The fixed delay to the start of the measurement is entered in this field.
	Value range:
	0 ms ... 1 s
	Resolution:
	1 ms
	Default:
	0 ms
"Max. Wait Interval"	The maximum duration of the test interval is entered in this field (maximum 20 samples).
	Value range:
	0 ms ... 10 s
	Resolution:
	1 ms (at a Max. Wait Interval of 1 ms a sample is taken every 50 µs)
	Default:
	0 ms
"Max. Wait Accuracy"	In this field, the maximum percentage difference between two measured values in succession (samples) is entered for test steps in which the measured value is applied (measurement with "Max. Wait Interval").
	Value range:
	0.0 % ... 10.0 %
	Default:
	1.0 %
"Average"	The number of measured values from which the mean for the measured result is to be determined is entered in this field.
	Value range:
	1 ... 1000
	Default:
	1
"Sample Interval"	The waiting time between the individual measurements for the formation of the mean is entered in this field (measurement with **"Average"**).
	Value range:
	5 µs ... 1 s
	Resolution:
	5 µs
	Default:
	5 µs (200 kHz)

5.5.3.5 Results Details

For the Diode test step, the following information is displayed in the **Results/Details** window after the execution of test step:
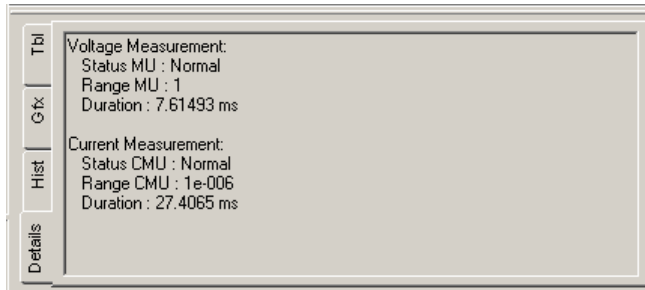
Figure 5-23: Test Step Diode, Results Details

	MU = voltage measuring unit
	CMU = current measuring unit
"Status"	Status of the measuring hardware:
	Normal, Overrange, Underrange, Max Wait Timeout.
"Range"	Range of the measuring hardware in which the measurement was carried out.
"Duration"	Duration of the last measurement carried out.

5.5.4 Discharge

ⓘ The discharging of capacitors is described in

5.5.4.1 General
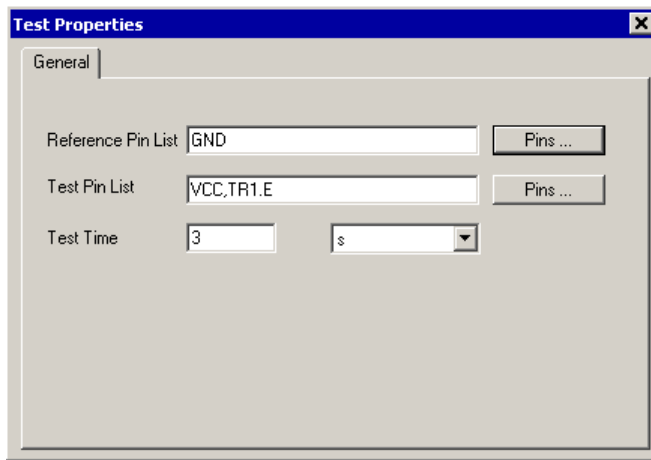
Figure 5-24: Test Step Discharge, Test Properties General

"Reference Pin List"	All measuring pins entered in the **"Test Pin List"** are connected one after the other to the measuring pins in the **"Reference Pin List"** via a discharge circuit.
"Test Pin List"	All measuring pins entered in the **"Test Pin List"** are connected one after the other to the measuring pins in the **"Reference Pin List"** via a discharge circuit.
"Test Time"	The maximum duration of the connection between the measuring pins from the **"Test Pin List"** and the **"Reference Pin List"** via the discharge circuit is entered in this field. The maximum duration for the discharge of all measuring pins is given. Value range: 1 ms ... 20 s Default: 3 s
Pins ...	Opens the **"Pins"** dialog box for inserting / removing pins on the Pin List. For more information, see section 5.5.12.2.

5.5.4.2 Results Details

For the Discharge test step, the following information is displayed in the Results/Details window after the execution of test step:
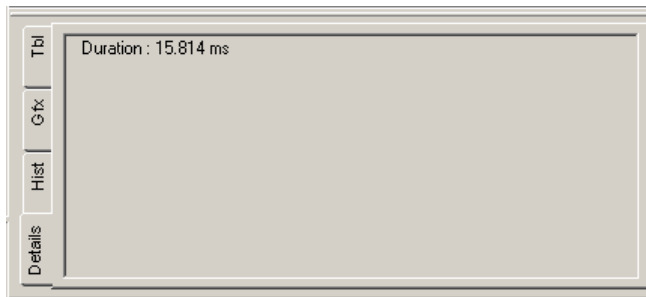
Figure 5-25: Test Step Discharge, Results Details

"Duration"	Duration of the last Discharge test step executed.

5.5.5 FET

The FET test is described in Chapter 10.7, "FET Test", on page 202.

5.5.5.1 Limits
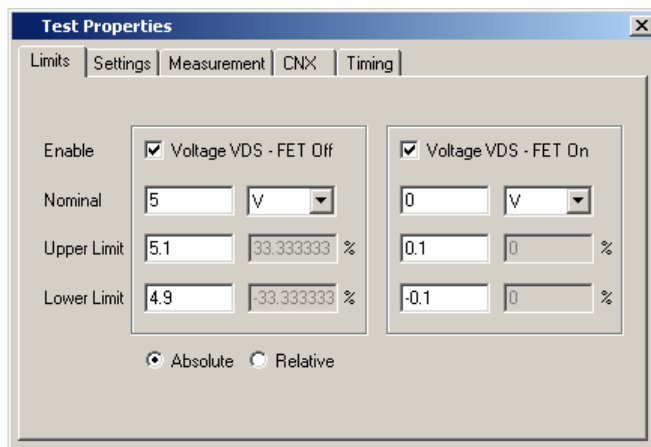
Figure 5-26: Test Step FET, Test Properties Limits

"Enable"	During the FET test, as an option two measurements are performed: • **"Voltage VDS - FET Off"** Measurement of the drain-source voltage of the FET while the FET is in switched off state. • **"Voltage VDS - FET On"** Measurement of the drain-source voltage of the FET while the FET is in switched on state.
	The related measurement is performed when the check box is activated. At least one measurement must be activated.

Voltage VDS - FET Off	
"Nominal"	The nominal value for the drain-source voltage of the switched off FET is entered in this field.
"Upper Limit" **"Lower Limit"**	The upper and lower limits of the measured drain-source voltage can be entered in these fields. The fields on the left contain the absolute value in the same unit as the nominal value. The fields on the right contain the deviation from the nominal value as a percentage.
	Inactive input fields (which appear in gray) are recalculated and displayed with the command **"Edit… Apply"**.
Voltage VDS - FET On	
"Nominal"	The nominal value for the drain-source voltage of the switched on FET is entered in this field.
"Upper Limit" **"Lower Limit"**	The upper and lower limits of the measured drain-source voltage can be entered in these fields. The fields on the left contain the absolute value in the same unit as the nominal value. The fields on the right contain the deviation from the nominal value as a percentage.
	Inactive input fields (which appear in gray) are recalculated and displayed with the command **"Edit… Apply"**.
"Absolute / Relative"	This setting determines whether limits will be entered as absolute values or relative values.

5.5.5.2 Settings
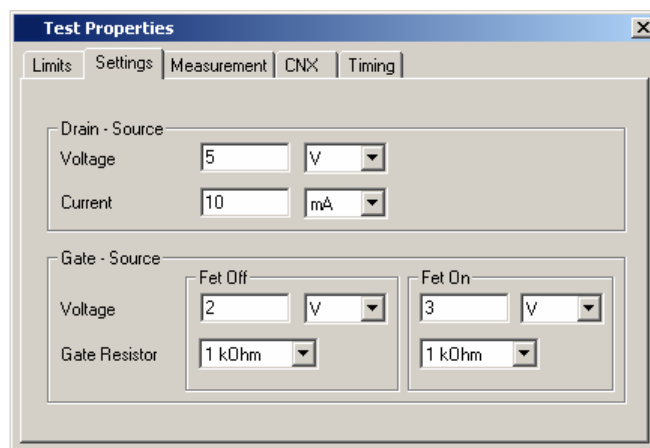
Figure 5-27: Test Step FET, Test Properties Settings

Drain - Source	
"Voltage"	The value of the TS-PSAM DCS voltage source between drain and source of the FET is entered in this field. Value range: -5.0 V … 5.0 V Default: 5.0 V
"Current"	The current limit of the TS-PSAM DCS voltage source is entered in this field. Value range: 3.0 µA … 100.0 mA Default: 10.0 mA
Gate - Source (FET Off and FET On)	
"Voltage"	The value of the TS-PICT AOS voltage source between gate and source of the FET is entered in this field. Value range: -5.0 V … 5.0 V Default: FET Off: 2.0 V FET On: 3.0 V
"Gate Resistor"	The value of the TS-PICT AOS output resistance which is connected to the gate of the FET is entered in this field. Value range: 0.0 Ω … 1.0 kΩ Default: 1.0 kΩ
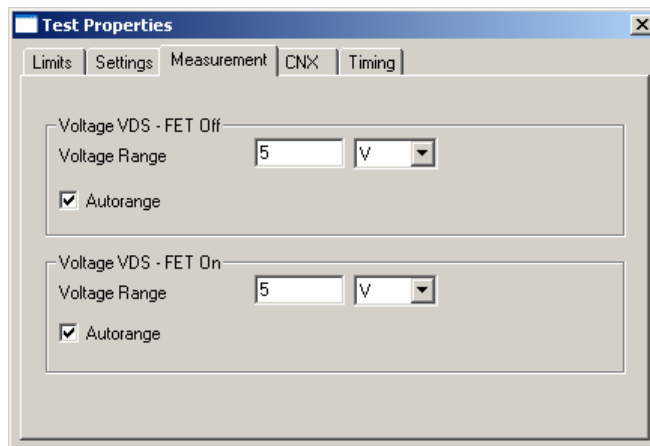
5.5.5.3 Measurement

Figure 5-28: Test Step FET, Test Properties Measurement

Voltage VDS-FET Off and *Voltage VDS-FET On*	
"Voltage Range"	The measuring range of the TS-PSAM voltmeter is entered in this field. Value range: 10.0 mV … 200.0 V Default: 5.0 V
"Autorange"	The measuring range for the voltmeter is set automatically when this check box is selected.

5.5.5.4 CNX
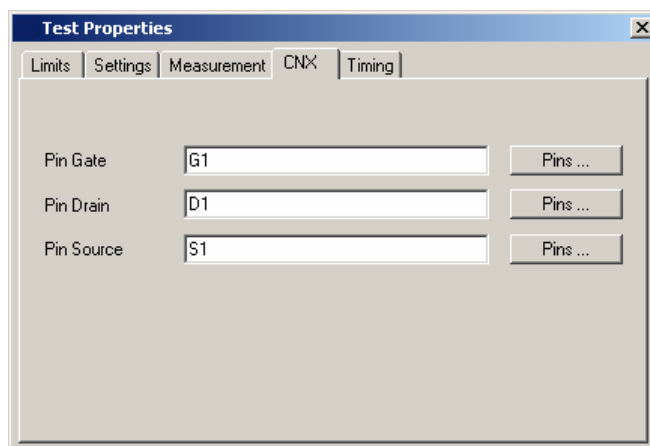
Figure 5-29: Test Step FET, Test Properties CNX

"Pin Gate"	The **gate pin** (gate of the FET) is entered in this field. Only one measuring pin can be entered.
"Pin Drain"	The **drain pin** (drain of the FET) is entered in this field. Only one measuring pin can be entered.
"Pin Source"	The **source pin** (source of the FET) is entered in this field. Only one measuring pin can be entered.
Pins ...	Opens the **"Pins"** dialog box for inserting / removing pins on the Pin List. For more information, see section 5.5.12.2.

5.5.5.5 Timing

Figure 5-30: Test Step FET, Test Properties Timing

ⓘ The individual options for the timing measurement operation are described in more detail in Chapter 5.5.12.1, "Timing", on page 103.

Voltage VDS-FET Off and *Voltage VDS-FET On*	
"Delay"	The fixed delay to the start of the measurement is entered in this field. Value range: 0 ms ... 1 s Resolution: 1 ms Default: 0 ms

"Max. Wait Interval"	The maximum duration of the test interval is entered in this field (maximum 20 samples).
	Value range:
	0 ms ... 10 s
	Resolution:
	1 ms
	Default:
	0 ms
"Max. Wait Accuracy"	In this field, the maximum percentage difference between two measured values in succession (samples) is entered for test steps in which the measured value is applied (measurement with "Max. Wait Interval").
	Value range:
	0.0 % ... 10.0 %
	Default:
	1.0 %
"Average"	The number of measured values from which the mean for the measured result is to be determined is entered in this field.
	Value range:
	1 ... 1000
	Default:
	1
"Sample Interval"	The waiting time between the individual measurements for the formation of the mean is entered in this field (measurement with Average).
	Value range:
	5 µs ... 1 s
	Resolution:
	5 µs
	Default:
	5 µs (200 kHz)

5.5.5.6 Results Details

For the FET test step, the following information is displayed in the Results/Details window after the execution of the test step:
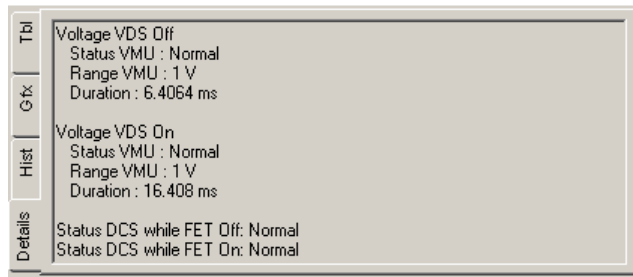
Figure 5-31: Test Step FET, Results Details

VMU = Voltage measuring unit (TS-PSAM)
DCS = DC Source (TS-PSAM)

Voltage VDS Off and *Voltage VDS On*	
"Status VMU"	Status of the voltage measurement unit: Normal, Overrange, Underrange, Max Wait Timeout
"Range VMU"	Range of the voltage measurement unit, in which the measurement was carried out.
"Duration"	Duration of the last measurement carried out.
"Status DCS"	Shows whether the TS-PSAM DCS voltage source is in the expected state. At the FET Off measurement the DCS must be in constant voltage state and at the FET On measurement the DCS must be in constant current state. If the DCS is in the expected state 'Normal' is displayed.

5.5.6 Impedance

The impedance test is described in Chapter 10.8, "Impedance Test", on page 203.

5.5.6.1 Limits

Figure 5-32: Test Step Impedance, Test Properties Limits

"Measurement"	The type of measured result and the related unit are selected in this drop-down list box. The internal calculation of the measured result is performed as a function of the measuring type selected (see also section Chapter 5.5.6.5, "Determination of Measured Value", on page 72).
"Nominal"	The nominal value for the impedance to be measured is entered in this field.
"Upper Limit" " Lower Limit"	The upper and lower limits of the measured impedance can be entered in these fields. The fields on the left contain the absolute value in the same unit as the nominal value. The fields on the right contain the deviation from the nominal value as a percentage. Inactive input fields (which appear in gray) are recalculated and displayed with the command **"Edit…Apply"**.
"Absolute / Relative"	This setting determines whether limits will be entered as absolute values or relative values.
"Correction"	A correction value can be entered in this field. The unit of the nominal value is assumed for that entry. The correction value is subtracted from the measurement. The result of the subtraction is compared with the limits. The correction value is handled together with the limits. This makes it possible to load, import and export it with the limits. See Chapter 5.6.1.7, "Menu Command <Limits>", on page 114 how to do that.

ⓘ The value ranges for Nominal, Upper Limit and Lower Limit are defined by the measuring and stimulation properties of the measuring hardware available. The related information is to be found in the user manuals and the data sheets on the measuring hardware.

For the 2-wire measurements, it is possible to specify a common correction value for all DUTs of a bench in the Application Layer Configuration File. For details, see Chapter 3.2.3, "Entries in APPLICATION.INI", on page 22.

5.5.6.2 Method
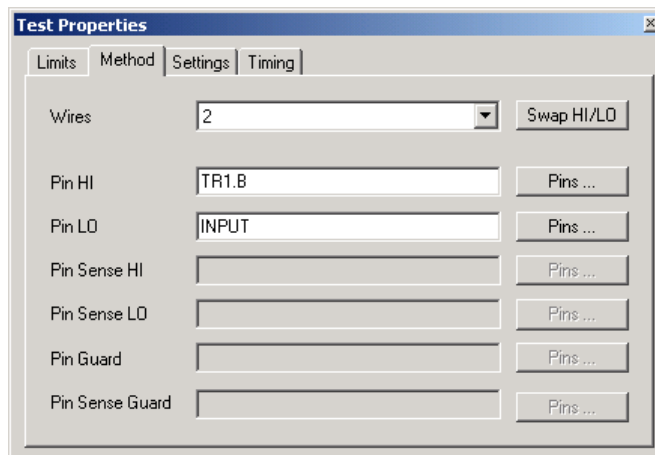
Figure 5-33: Test Step Impedance, Test Properties Method

"Wires"	The impedance measuring method is selected in this drop-down list box. The measuring methods are described in Chapter 10.9, "Resistor Test", on page 208.
"Pin HI"	The **Pin HI** for all measuring methods is entered in this field. Only one measuring pin can be entered.
"Pin LO"	The **Pin LO** for all measuring methods is entered in this field. Only one measuring pin can be entered.
"Pin Sense HI"	The **Pin Sense HI** is entered in this field for the following measuring methods: • 4-wire measuring method • 6-wire guarded measuring method Only one measuring pin can be entered.
"Pin Sense LO"	The **Pin Sense LO** is entered in this field for the following measuring methods: • 4-wire measuring method • 6-wire guarded measuring method Only one measuring pin can be entered.

"Pin Guard"	The **Pin Guard** is entered in this field for the following measuring methods: • 3-wire guarded measuring method • 4-wire guarded measuring method • 6-wire guarded measuring method Several measuring pins can be entered.
"Pin Sense Guard"	The **Pin Sense Guard** is entered in this field for the following measuring methods: • 4-wire guarded measuring method • 6-wire guarded measuring method Only one measuring pin can be entered.
Pins ...	Opens the **Pins** dialog box for inserting / removing pins on the Pin List. For more information, see Chapter 5.5.12.2, "Editing Pin Lists", on page 105.
Swap HI/LO	Exchanges the contents of the Pin HI / Pin LO and Pin Sense HI / Pin Sense LO edit fields.

Figure 5-34: R&S TS-PMB wiring

As can be seen in Figure 5-34, only the following wiring between analog bus and UUT is possible for the R&S TS-PMB Matrix Module B:

Analog Bus	Pin
A1	P1, P3, P5, P7, P9, ... , P87, P89
A2	P2, P4, P6, P8, P10, ... , P88, P90
B1	P1, P3, P5, P7, P9, ... , P87, P89
B2	P2, P4, P6, P8, P10, ... , P88, P90
C1	P1, P3, P5, P7, P9, ... , P87, P89
C2	P2, P4, P6, P8, P10, ... , P88, P90
D1	P1, P3, P5, P7, P9, ... , P87, P89
D2	P2, P4, P6, P8, P10, ... , P88, P90

For guarded 6-wire impedance measurement, please note that the wiring is to be done in the following way:

● The Pins HI and Sense HI must be complementary, i.e. HI even and Sense HI odd or HI odd and Sense HI even.
● The Pins LO and Sense LO must also be complementary.

- For the Pin Guard and Sense Guard, there are no limitations. Guard is a pin list and can contain a combination of even and odd pins.

Example:

Pin HI	even
Pin Sense HI	odd
Pin LO	odd
Pin Sense LO	even
Pin Guard	even, odd...
Pin Sense Guard	odd

If this even/odd rule is not adhered to, the Enhanced Generic Test Software Library R&S EGTSL will report a compile error.

During generation of an ICT program by the Automatic Test Generator ATG this even/odd rule is followed.

5.5.6.3 Settings
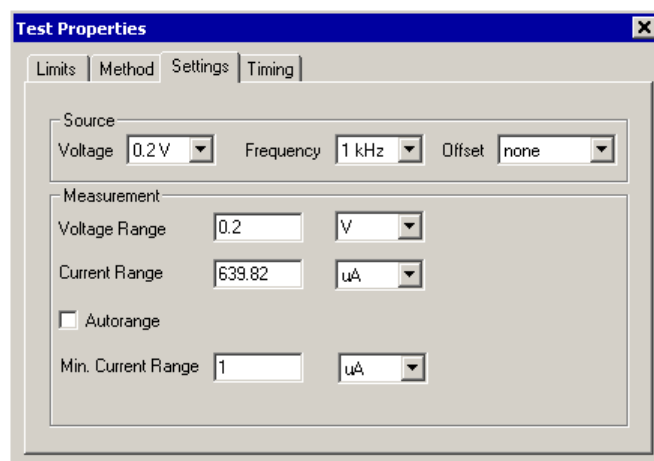
Figure 5-35: Test Step Impedance, Test Properties Settings

Source	
"Voltage"	The voltage of the signal source is selected in this drop-down list box. Value range: 0.1 V, 0.2 V, 1.0 V Default: 0.2 V

"Frequency"	The frequency of the signal source is selected from this drop-down list box.
	Value range:
	100 Hz, 1 kHz, 10 kHz
	Default:
	10 kHz
"Offset"	The offset of the signal source is selected from this drop-down list box.
	Value range:
	none, positive, negative
	Default:
	none
Measurement	
"Voltage Range"	The measuring range of the voltmeter is entered in this field.
	Value range:
	10 mV ... 5 V
	Default:
	0.2 V
"Current Range"	The measuring range of the current meter is entered in this field.
	Value range:
	1 µA ... 100 mA
	Default:
	100 mA
"Autorange"	The measuring ranges of the voltmeter and current meter are set automatically when this check box is selected.
"Min. Current Range"	The smallest permissible measuring range for the current meter on the use of the Autorange function is entered in this field.
	Value range:
	1 µA ... 200 mA
	Default:
	2 µA

In some measuring conditions, the smallest measuring ranges are unstable. More reliable measured results are obtained in larger measuring ranges.

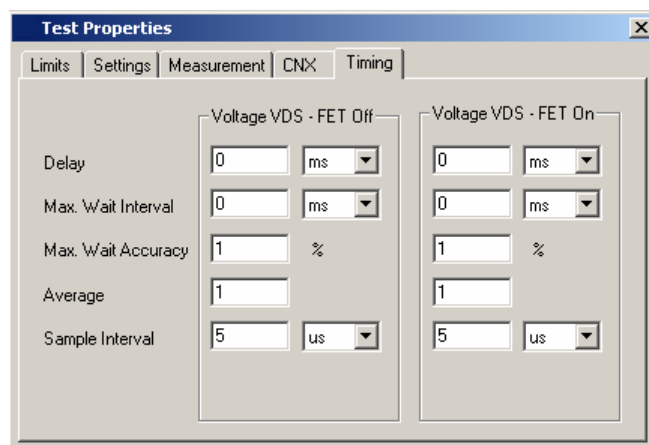### 5.5.6.4 Timing
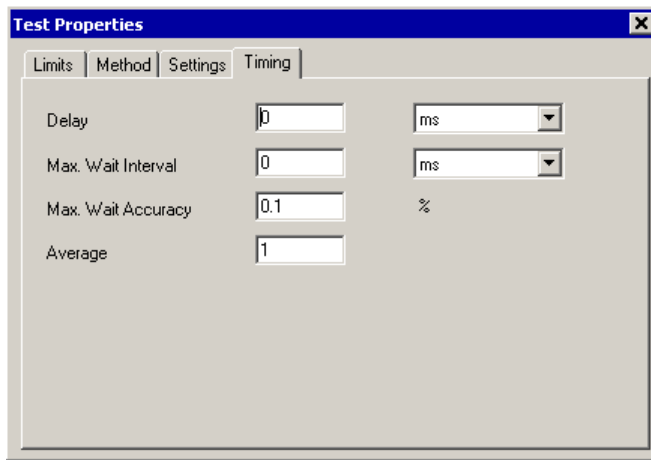
Figure 5-36: Test Step Impedance, Test Properties Timing

ⓘ The individual options for the timing of the measurement operation are described in more detail in Chapter 5.5.12.1, "Timing", on page 103.

"Delay"	The fixed delay to the start of the measurement is entered in this field.
	Value range:
	0 ms ... 1 s
	Resolution:
	1 ms
	Default:
	0 ms
"Max. Wait Interval"	The maximum duration of the test interval is entered in this field (maximum 20 samples).
	Value range:
	0 ms ... 10 s
	Resolution:
	1 ms
	Default:
	0 ms

"Max. Wait Accuracy"	In this field, the maximum percentage difference between two measured values in succession (samples) is entered for test steps in which the measured value is applied (measurement with "Max. Wait Interval").
	Value range:
	0.0 % ... 10.0 %
	Default:
	1.0 %
"Average"	The number of signal periods to be measured from which a mean is formed for the measured result is entered in this field.
	Value range:
	1 ... 1000
	Default:
	1

5.5.6.5 Determination of Measured Value
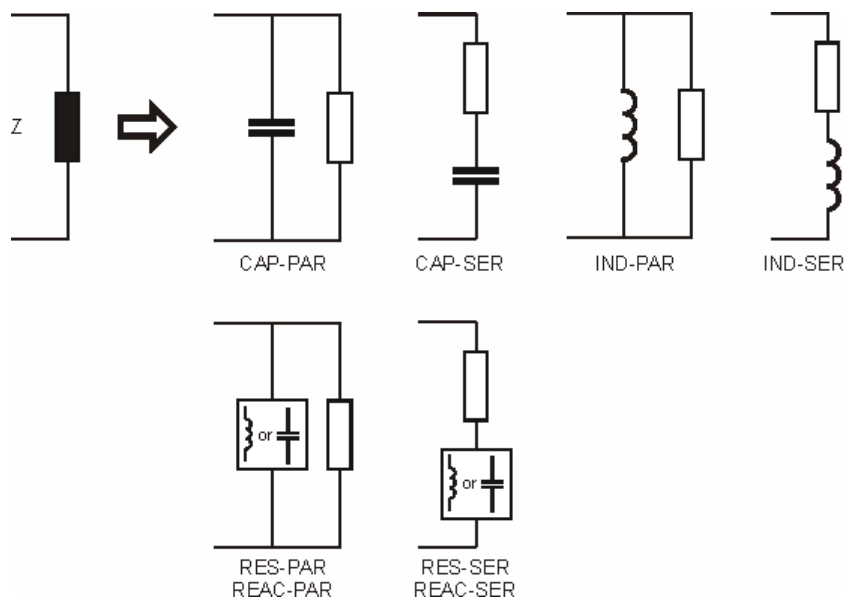
Figure 5-37: Equivalent circuit diagrams for determination of measured value

An equivalent circuit **must** be entered for the calculation of the required measured result for the capacitance measurement and the inductance measurement (see Figure 5-37). The selection is made using the **"Measurement"** drop-down list box in Test Properties Limits.

Table 5-1: Determination of measured value (1/2)

Measurement	Measured result	Equivalent circuit	Unit
CAP-PAR	Capacitance	Capacitance measurement based on a parallel equivalent circuit	Farad (F)
CAP-SER	Capacitance	Capacitance measurement based on a serial equivalent circuit	Farad (F)
IND-PAR	Inductance	Inductance measurement based on a parallel equivalent circuit	Henry (H)
IND-SER	Inductance	Inductance measurement based on a serial equivalent circuit	Henry (H)

Two different representations of the impedance are available for the measurement of the impedance. On the one hand the representation in real and imaginary parts, on the other hand the representation in magnitude and phase. An equivalent circuit **must** be entered for the calculation of the required measured result for the representation in real and imaginary parts (see Figure 5-37).

Table 5-2: Determination of measured value (2/2)

Measurement	Measured result	Equivalent circuit	Unit
RES-SER	Real part of the measured impedance	Impedance measurement based on a serial equivalent circuit	Ohm (Ω)
RES-PAR	Real part of the measured impedance	Impedance measurement based on a parallel equivalent circuit	Ohm (Ω)
REAC-SER	Imaginary part of the measured impedance	Impedance measurement based on a serial equivalent circuit	Ohm (Ω)
REAC-PAR	Imaginary part of the measured impedance	Impedance measured	Ohm (Ω)
IMP-MAG	Magnitude of the impedance measured		Ohm (Ω)
IMP-PHASE	Phase of the measured impedance		deg

5.5.6.6 Results Details

For the Impedance test step, the following information is displayed in the **"Results/ Detail"s** window after the execution of test step:
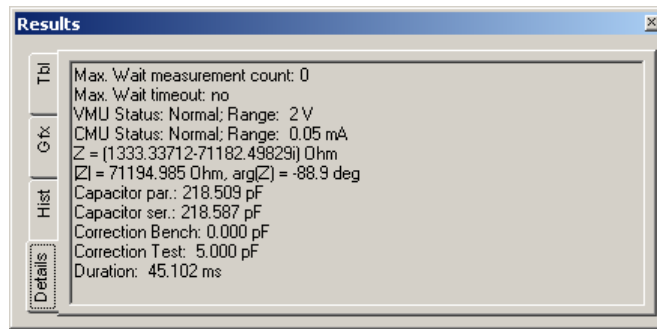
Figure 5-38: Test Step Impedance, Results Details

| Max. Wait measurement count	Number of the measurements which were carried out with Max.Wait until the measured value was stable.
Max. Wait timeout	"Yes", if the measured value was not stable even after 20 Max. Wait measurements.
VMU Status	Status of the voltage measuring unit: Normal, Overrange, Underrange
VMU Range	Range of the voltage measuring unit in which the measurement was carried out.
CMU Status	Status of the current measuring unit: Normal, Overrange, Underrange
CMU Range	Range of the current measuring unit in which the measurement was carried out.
Z	Real part and imaginary part of the impedance
|Z|, arg(Z)	Magnitude and Phase of the impedance
Capacitor par.	Measured value for the parallel equivalent circuit diagram
Capacitor ser.	Measured value for the serial equivalent circuit diagram
Correction Bench	Bench-specific correction value taken into account for calculating the result. The indicated value is specified in the Application Layer Configuration File. For details, see Chapter 3.2.3, "Entries in APPLICATION.INI", on page 22section .
Correction Test	Test-specific correction value taken into account for calculating the result. The indicated value is specified in the test properties limits.
Duration	Duration of the last Impedance test step carried out.

5.5.7 Resistor

ⓘ The resistor test is described in Chapter 10.9, "Resistor Test", on page 208.

5.5.7.1 Limits

Figure 5-39: Test Step Resistor, Test Properties Limits

"Nominal"	The nominal value for the resistor to be measured is entered in this field.
"Upper Limit" "Lower Limit"	The upper and lower limits of the measured resistor can be entered in these fields. The fields on the left contain the absolute value in the same unit as the nominal value. The fields on the right contain the deviation from the nominal value as a percentage. Inactive input fields (which appear in gray) are recalculated and displayed with the command **"Edit… Apply"**.
"Absolute / Relative"	This setting determines whether limits will be entered as absolute values or relative values.
"Correction"	A correction value can be entered in this field. The unit of the nominal value is assumed for that entry. The correction value is subtracted from the measurement. The result of the subtraction is compared with the limits. The correction value is handled together with the limits. This makes it possible to load, import and export it with the limits. See Chapter 5.6.1.7, "Menu Command <Limits>", on page 114 how to do that.

ⓘ The value ranges for Nominal, Upper Limit and Lower Limit are defined by the measuring and stimulation properties of the measuring hardware available. The related information is to be found in the user manuals and the data sheets on the measuring hardware.

For the 2-wire resistor measurements, it is possible to specify a common correction value for all DUTs of a bench in the Application Layer Configuration File. For details, see Chapter 3.2.3, "Entries in APPLICATION.INI", on page 22.

5.5.7.2 Method
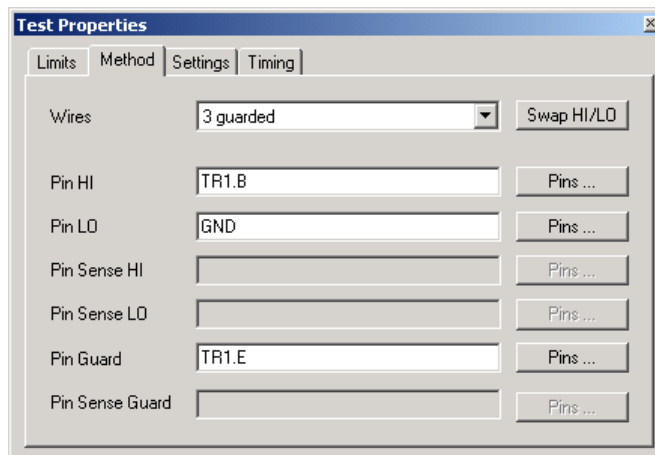
Figure 5-40: Test Step Resistor, Test Properties Method

"Wires"	The resistance measuring method is selected in this drop-down list box. The measuring methods are described in Chapter 10.9, "Resistor Test", on page 208.
"Pin HI"	The **Pin HI** for all measuring methods is entered in this field. Only one measuring pin can be entered.
"Pin LO"	The **Pin LO** for all measuring methods is entered in this field. Only one measuring pin can be entered.
"Pin Sense HI"	The **Pin Sense HI** is entered in this field for the following measuring methods: • 4-wire measuring method • 6-wire guarded measuring method Only one measuring pin can be entered.
"Pin Sense LO"	The **Pin Sense LO** is entered in this field for the following measuring methods: • 4-wire measuring method • 6-wire guarded measuring method Only one measuring pin can be entered.

"Pin Guard"	The **Pin Guard** is entered in this field for the following measuring methods: • 3-wire guarded measuring method • 4-wire guarded measuring method • 6-wire guarded measuring method Several measuring pins can be entered.
"Pin Sense Guard"	The **Pin Sense Guard** is entered in this field for the following measuring methods: • 4-wire guarded measuring method • 6-wire guarded measuring method Only one measuring pin can be entered.
Pins ...	Opens the **Pins** dialog box for inserting / removing pins on the Pin List. For more information, see Chapter 5.5.12.2, "Editing Pin Lists", on page 105.
Swap HI/LO	Exchanges the contents of the Pin HI / Pin LO and Pin Sense HI / Pin Sense LO edit fields.

Figure 5-41: TS-PMB

As can be seen in Figure 5-41, only the following wiring between analog bus and UUT is possible for the TS-PMB Matrix Module B:

Analog Bus	Pin
A1	P1, P3, P5, P7, P9, ... , P87, P89
A2	P2, P4, P6, P8, P10, ... , P88, P90
B1	P1, P3, P5, P7, P9, ... , P87, P89
B2	P2, P4, P6, P8, P10, ... , P88, P90
C1	P1, P3, P5, P7, P9, ... , P87, P89
C2	P2, P4, P6, P8, P10, ... , P88, P90
D1	P1, P3, P5, P7, P9, ... , P87, P89
D2	P2, P4, P6, P8, P10, ... , P88, P90

For guarded 6-wire resistance measurement, the wiring is to be done in the following way:

● The Pins HI and Sense HI must be complementary, i.e. HI even and Sense HI odd or HI odd and Sense HI even.

● The Pins LO and Sense LO must also be complementary.

- For the Pin Guard and Sense Guard, there are no limitations.
  Guard is a pin list and can contain a combination of even and odd pins.

Example:

Pin HI	even
Pin Sense HI	odd
Pin LO	odd
Pin Sense LO	even
Pin Guard	even, odd...
Pin Sense Gurard	odd

If this even/odd rule is not adhered to, the Enhanced Generic Test Software Library R&S EGTSL will report a compile error.

During generation of an ICT program by the Automatic Test Generator ATG this even/odd rule is followed.

5.5.7.3 Settings
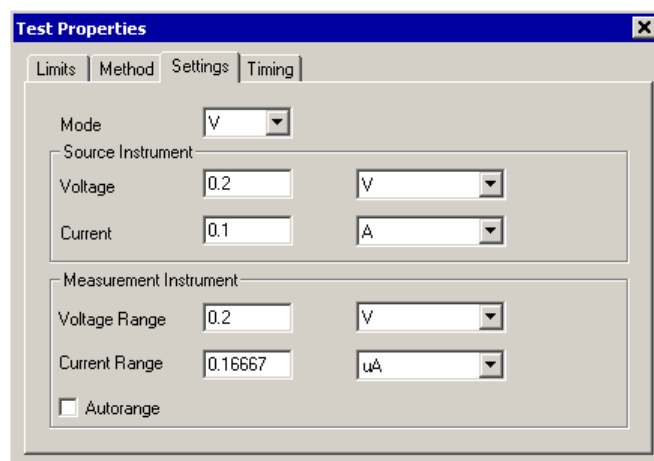
Figure 5-42: Test Step Resistor, Test Properties Settings

"Mode"	The type of resistor measurement is selected from this drop-down list box:
	"V": application of voltage with current measurement
	" C": application of current with voltage measurement
Source Instrument	

"Voltage"	The voltage of the voltage source is entered in this field. Value range: 1 mV ... 5 V Default: 0.2 V
"Current"	The current limit of the voltage source is entered in this field. Value range: 1 µA ... 100 mA Default: 100 mA
Measurement Instrument	
"Voltage Range"	The measuring range of the voltmeter is entered in this field. Value range: 10 mV ... 5 V Default: 0.2 V
"Current Range"	The measuring range of the current meter is entered in this field. Value range: 0 µA ... 200 mA Default: 100 mA
"Autorange"	The measuring ranges for the voltmeter and current meter are set automatically when this check box is selected.

Only one measuring range is relevant depending on the type of resistor measurement:

Mode V:	Current Range (application of voltage with current measurement)
Mode C:	Voltage Range (application of current with voltage measurement)

When Autorange is selected, the search for the optimal measuring range starts with the values entered in the **"Voltage Range"** (voltage measurement) and **"Current Range"** (current measurement) fields.

5.5.7.4 Timing
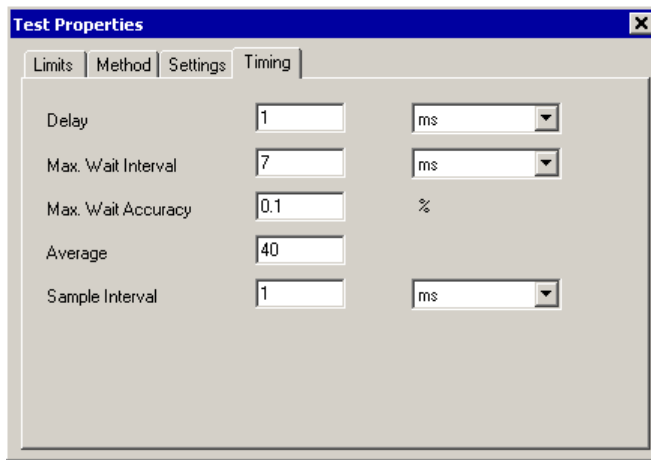
Figure 5-43: Test Step Resistor, Test Properties Timing

ⓘ The individual options for the timing of the measurement operation are described in more detail in Chapter 5.5.12.1, "Timing", on page 103.

"Delay"	The fixed delay to the start of the measurement is entered in this field. Value range: 0 ms ... 1 s Resolution: 1 ms Default: 0 ms
"Max. Wait Interval"	The maximum duration of the test interval is entered in this field (maximum 20 samples). Value range: 0 ms ... 10 s Resolution: 1 ms (at a Max. Wait Interval of 1 ms a sample is taken every 50 µs) Default: 0 ms
"Max. Wait Accuracy"	In this field, the maximum percentage difference between two measured values in succession (samples) is entered for test steps in which the measured value is applied (measurement with "Max. Wait Interval"). Value range: 0.0 % ... 10.0 % Default: 1.0 %

"Average"	The number of measured values from which the mean for the measured result is to be determined is entered in this field.
	Value range:
	1 ... 1000
	Default:
	1
"Sample Interval"	The waiting time between the individual measurements for the formation of the mean is entered in this field (measurement with Average).
	Value range:
	5 µs ... 1 s
	Resolution:
	5 µs
	Default:
	5 µs (200 kHz)

5.5.7.5 Results Details

For the Resistor test step, the following information is displayed in the **"Results/ Details"** window after the execution of test step:
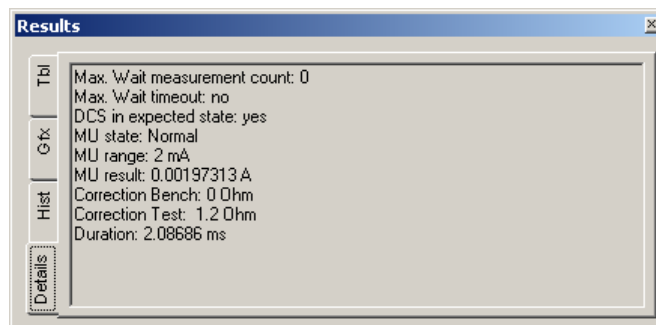
Figure 5-44: Test Step Resistor, Results Details

Max. Wait measurement count	Number of the measurements which were carried out with Max.Wait until the measured value was stable.
Max. Wait timeout	"Yes", if the measured value was not stable even after 20 Max. Wait measurements.
DCS in expected state	Shows whether the voltage source is in the expected status. In Mode C it must be in the current limit, in Mode V it does not have to be in the current limit.
MU State	Status of the voltage measuring unit (Mode C) or of the current measuring unit (Mode V): Normal, Overrange, Underrange
MU Range	Voltage or current range, in which the measurement was carried out.

MU Result	Result of the voltage measurement (Mode C) or current measurement (Mode V)
Correction Bench	Bench-specific correction value taken into account for calculating the result. The indicated value is specified in the Application Layer Configuration File. See section 3.2.3 for details.
Correction Test	Test-specific correction value taken into account for calculating the result. The indicated value is specified in the test properties limits.
Duration	Duration of the last Resistor test step carried out

5.5.8 Short

ⓘ The short-circuit test is described in Chapter 10.10, "Short-circuit Test", on page 212.

5.5.8.1 General
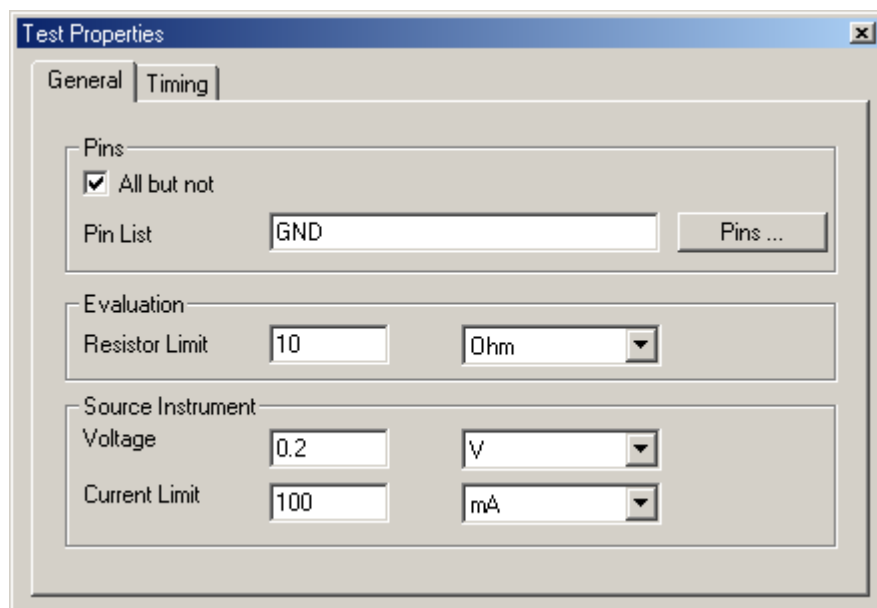
Figure 5-45: Test Step Short, Test Properties General

Pins	
"All but not"	All available measuring pins are checked for short-circuit when the check box is selected. The list of available measuring pins is read from the **"Available Pins"** (see Chapter 5.5.12.2, "Editing Pin Lists", on page 105). The measuring pins entered in the **"Pin List"** are not checked for short-circuit.

"Pin List"	**"All but not"** check box cleared:
	All measuring pins entered in the **"Pin List"** are checked for short-circuit.
	"All but not" check box selected:
	all pins except for the measuring pins entered in the Pin List are checked for short-circuit.
Pins ...	Opens the **Pins** dialog box for inserting / removing pins on the Pin List. For more information, see Chapter 5.5.12.2, "Editing Pin Lists", on page 105.
Evaluation	
"Resistor Limit"	The maximum resistance that is still evaluated as a short-circuit is entered in this field.
	Value range:
	1 Ω ... 1 kΩ
	Default:
	10 Ω
Source Instrument	
"Voltage"	The voltage of the voltage source is entered in this field.
	Value range:
	0.1 V ... 0.5 V
	Default:
	0.2 V
"Current Limit"	The current limit of the voltage source is entered in this field.
	Value range:
	1 µA ... 100 mA
	Default:
	100 mA

When the **Voltage** value is appropriately selected, the threshold voltage can be designed such that diodes/transistors can be considered the same as missing components. Diode/transistors must not become forward biased.

5.5.8.2 Timing
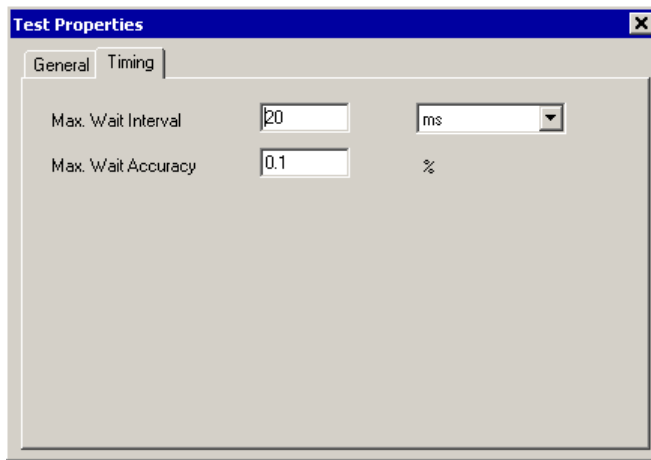
Figure 5-46: Test Step Short, Test Properties Timing

ⓘ The individual options for the timing of the measurement operation are described in more detail in Chapter 5.5.12.1, "Timing", on page 103.

"Max. Wait Interval"	The maximum duration of the test interval is entered in this field (maximum 20 samples).
	Value range:
	0 ms ... 10 s
	Resolution:
	1 ms (at a Max. Wait Interval of 1 ms a sample is taken every 50 µs)
	Default:
	20 ms
"Max. Wait Accuracy"	In this field, the maximum percentage difference between two measured values in succession (samples) is entered for test steps in which the measured value is applied (measurement with "Max. Wait Interval").
	Value range:
	0.0 % ... 10.0 %
	Default:
	1.0 %

5.5.8.3 Results Details

For the Short test step, the following information is displayed in the **"Results/Details"** window after the execution of test step:
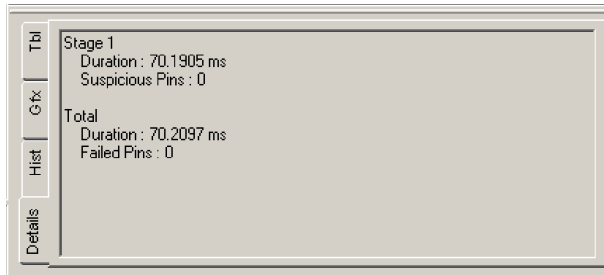
Figure 5-47: Test Step Short, Results Details

Stage 1:	
Duration	Duration of the 1st stage.
Suspicious Pins	Number of "suspicious" pins found in the first stage.
Total:	
Duration	Total duration of the measurement.
Failed Pins	Number of short-circuited pins.

5.5.9 Transistor

The transistor test is described in Chapter 10.11, "Transistor Test", on page 213.

5.5.9.1 Limits
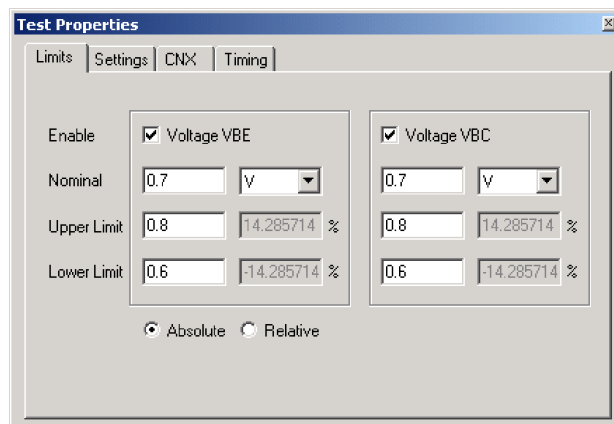
Figure 5-48: Test Step Transistor, Test Properties Limits

"Enable"	During the transistor test, as an option two measurements are performed: • **"Voltage VBE"** Measurement of the forward bias voltage (knee voltage) of the base-emitter diode • **"Voltage VBC"** Measurement of the forward bias voltage (knee voltage) of the base-collector diode The related measurement is performed when the check box is activated. At least one measurement must be activated.
"Nominal"	The nominal value for the forward bias voltage (knee voltage) of the base-emitter diode or base-collector diode is entered in this field. The nominal value and limits are positive for NPN transistors and negative for PNP transistors.
"Upper Limit" **"Lower Limit"**	The upper and lower limits of the measured forward bias voltage (knee voltage) of the base-emitter diode or base-collector diode can be entered in these fields. The fields on the left contain the absolute value in the same unit as the nominal value. The fields on the right contain the deviation from the nominal value as a percentage. Inactive input fields (which appear in gray) are recalculated and displayed with the command **"Edit…Apply"**.
"Absolute / Relative"	This setting determines whether limits will be entered as absolute values or relative values.

5.5.9.2 Settings
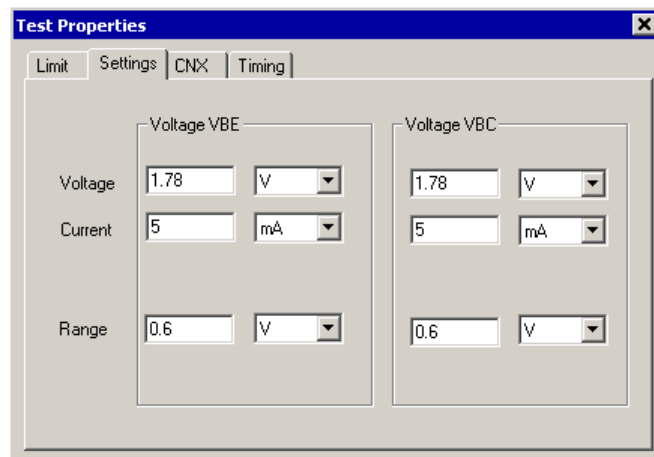
Figure 5-49: Test Step Transistor, Test Properties Settings

Voltage VBE + Voltage VBC	
"Voltage"	The voltage of the voltage source is entered in this field. The voltage is positive for NPN transistors and negative for PNP transistors. Value range: -5.0 V ... 5.0 V Default: 1.8 V
"Current"	The current limit of the voltage source is entered in this field. Value range: 1.0 μA ... 100.0 mA Default: 5.0 mA
"Range"	The measuring range of the voltmeter is entered in this field. Value range: 10.0 mV ... 5.0 V Default: 1.0 V

5.5.9.3 CNX

Figure 5-50: Test Step Transistor, Test Properties CNX

"Pin Base"	The **base pin** (base of the transistor) is entered in this field. Only one measuring pin can be entered.
"Pin Collector"	The **collector pin** (collector of the transistor) is entered in this field. Only one measuring pin can be entered.

"Pin Emitter"	The **emitter pin** (emitter of the transistor) is entered in this field. Only one measuring pin can be entered.
Pins ...	Opens the **Pins** dialog box for inserting / removing pins on the Pin List. For more information, see Chapter 5.5.12.2, "Editing Pin Lists", on page 105.

5.5.9.4 Timing
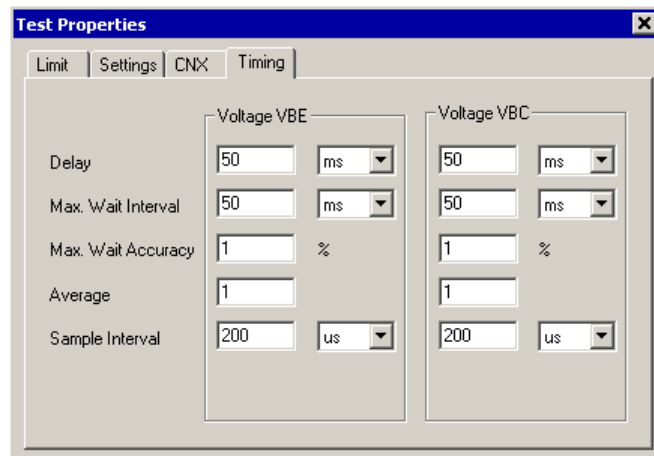
Figure 5-51: Test Step Transistor, Test Properties Timing

(i) The individual options for the timing of the measurement operation are described in more detail in Chapter 5.5.12.1, "Timing", on page 103.

Voltage VBE + Voltage VBC	
"Delay"	The fixed delay to the start of the measurement is entered in this field. Value range: 0 ms ... 1 s Resolution: 1 ms Default: 0 ms
"Max. Wait Interval"	The maximum duration of the test interval is entered in this field (maximum 20 samples). Value range: 0 ms ... 10 s Resolution: 1 ms Default: 0 ms

"Max. Wait Accuracy"	In this field, the maximum percentage difference between two measured values in succession (samples) is entered for test steps in which the measured value is applied (measurement with "Max. Wait Interval"). Value range: 0.0 % ... 10.0 % Default: 1.0 %
"Average"	The number of measured values from which the mean for the measured result is to be determined is entered in this field. Value range: 1 ... 1000 Default: 1
"Sample Interval"	The waiting time between the individual measurements for the formation of the mean is entered in this field (measurement with **"Average"**). Value range: 5 µs ... 1 s Resolution: 5 µs Default: 5 µs (200 kHz)

5.5.9.5 Results Details

For the Transistor test step, the following information is displayed in the **"Results/ Details"** window after the execution of test step:

Figure 5-52: Test Step Transistor, Results Details

	MU = Voltage measuring unit
"Status"	Status of the voltage measuring unit: Normal, Overrange, Underrange, Max Wait Timeout.

"Range"	Range of the voltage measuring unit, in which the measurement was carried out.
"Duration"	Duration of the last measurement carried out.

5.5.10 Transistor Beta

ⓘ The Transistor Beta test is described in

5.5.10.1 Limits
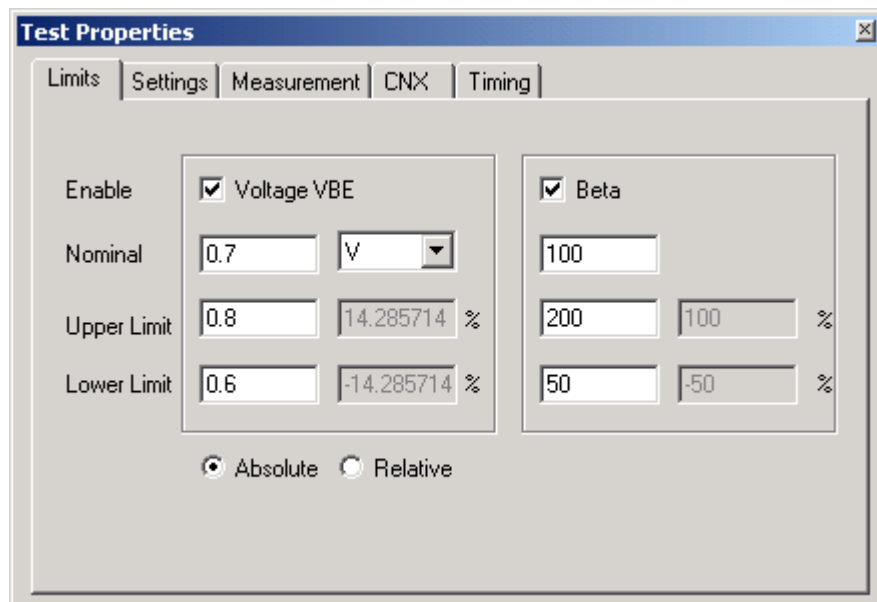
Figure 5-53: Test Step Transistor Beta, Test Properties Limits

"Enable"	During the transistor beta test, as an option two measurements are performed: • **"Voltage VBE"** Measurement of the forward bias voltage (knee voltage) of the base-emitter diode. • **"Beta"** Measurement of the dynamic current gain. The related measurement is performed when the check box is activated. At least one measurement must be activated.
Voltage VBE	
"Nominal"	The nominal value for the forward bias voltage (knee voltage) of the base-emitter diode is entered in this field.

"Upper Limit" **"Lower Limit"**	The upper and lower limits of the measured forward bias voltage can be entered in these fields. The fields on the left contain the absolute value in the same unit as the nominal value. The fields on the right contain the deviation from the nominal value as a percentage. Inactive input fields (which appear in gray) are recalculated and displayed with the command **"Edit…Apply"**.
Beta	
"Nominal"	The nominal value for the current gain (beta) is entered in this field.
"Upper Limit" **"Lower Limit"**	The upper and lower limits of the measured current gain (beta) can be entered in these fields. The fields on the left contain the absolute value in the same unit as the nominal value. The fields on the right contain the deviation from the nominal value as a percentage. Inactive input fields (which appear in gray) are recalculated and displayed with the command **"Edit…Apply"**.
"Absolute / Relative"	This setting determines whether limits will be entered as absolute values or relative values.

5.5.10.2 Settings
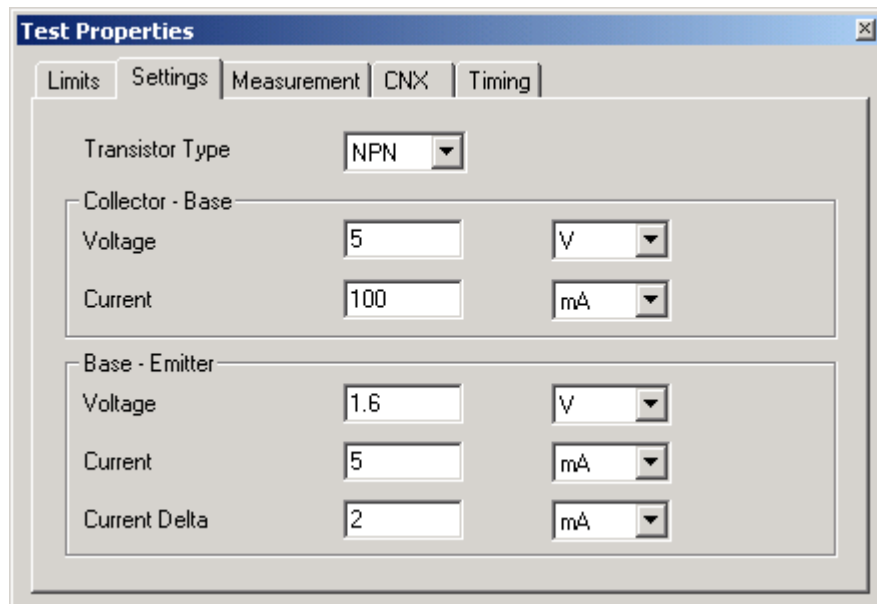
Figure 5-54: Test Step Transistor Beta, Test Properties Settings

" Transistor Type"	The type of the transistor (NPN or PNP).
	Note that the voltage values are always entered as positive numbers.
	The actual polarity is determined by the transistor type.
Collector - Base	
"Voltage"	The voltage of the R&S TS-PSU / R&S TS-PSU12 voltage source between collector and base is entered in this field.
	Value range:
	0.0 V … 10.0 V
	Default:
	5.0 V
"Current"	The current limit of the R&S TS-PSU / R&S TS-PSU12 voltage source is entered in this field.
	Value range:
	1.0 µA … 100.0 mA
	Default:
	100.0 mA
Base - Emitter	
"Voltage"	The voltage of the DCS voltage source between base and emitter is entered in this field.
	Value range:
	0.0 V … 5.0 V
	Default:
	1.6 V
"Current"	The current limit of the DCS voltage source is entered in this field.
	Value range:
	1.0 µA … 100.0 mA
	Default:
	5.0 mA
"Current Delta"	The increase of the current limit of the DCS voltage source between the first and second current measurement is entered in this field.
	Value range:
	-100.0 mA … 100.0 mA
	Default:
	2.0 mA

ⓘ　The sum of Current and Current Delta must be in the range 0 mA … 100 mA.

5.5.10.3 Measurement
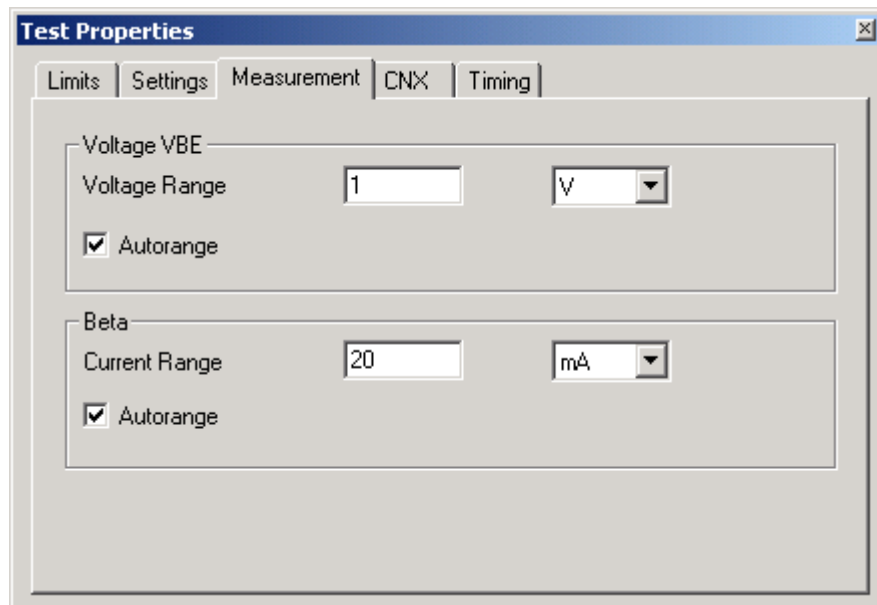
Figure 5-55: Test Step Transistor Beta, Test Properties Measurement

Voltage VBE	
"Voltage Range"	The measuring range of the voltmeter is entered in this field. Value range: 10.0 mV … 100.0 V Default: 1.0 V
"Autorange"	The measuring range for the voltmeter is set automatically when this check box is selected.
Beta	
"Current Range"	The measuring range of the current meter is entered in this field. Value range: 1.0 µA … 100.0 mA Default: 20.0 mA
"Autorange"	The measuring range for the current meter is set automatically when this check box is selected.

5.5.10.4 CNX
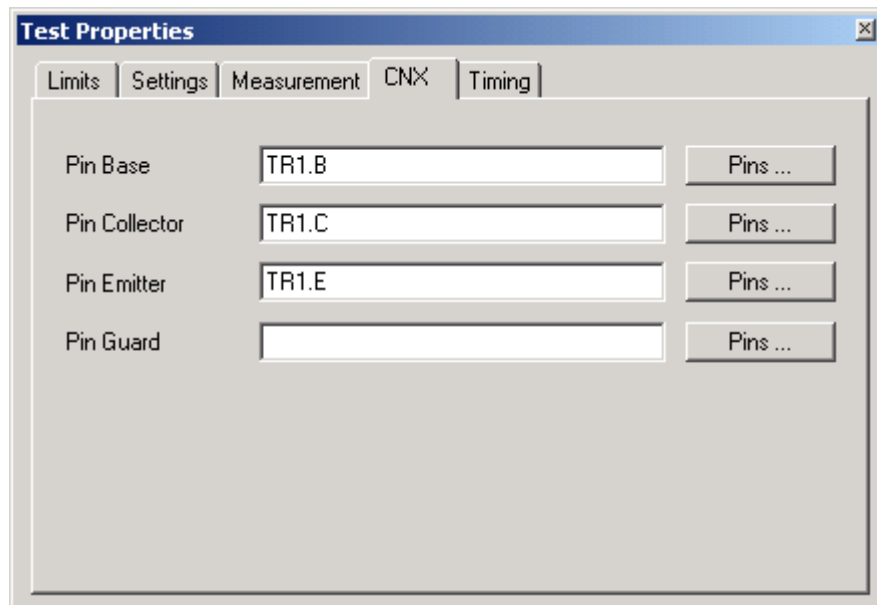
Figure 5-56: Test Step Transistor Beta, Test Properties CNX

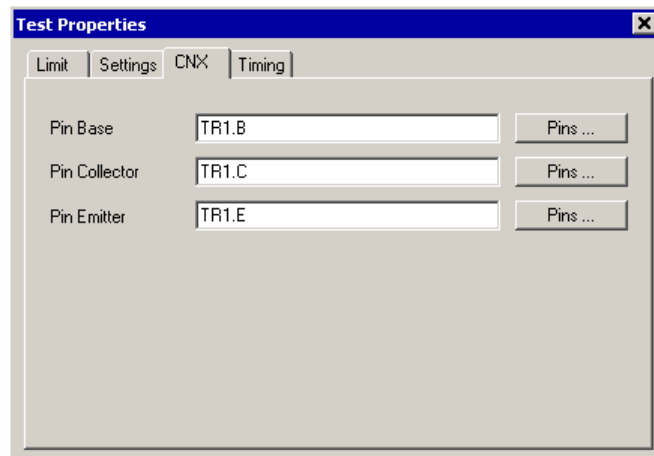| "Pin Base"	The **base pin** (base of the transistor) is entered in this field. Only one measuring pin can be entered.
"Pin Collector"	The **collector pin** (collector of the transistor) is entered in this field. Only one measuring pin can be entered.
"Pin Emitter"	The **emitter pin** (emitter of the transistor) is entered in this field. Only one measuring pin can be entered.
"Pin Guard"	The **guard pins** are entered in this field. Several measuring pins can be entered. The field remains empty for unguarded measurements.
Pins ...	Opens the **Pins** dialog box for inserting / removing pins on the Pin List. For more information, see Chapter 5.5.12.2, "Editing Pin Lists", on page 105.

5.5.10.5 Timing
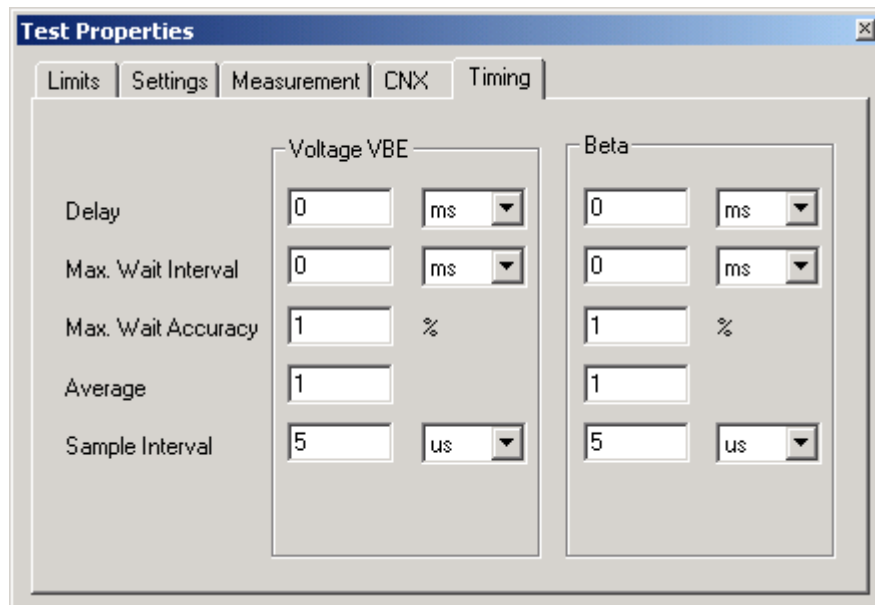
Figure 5-57: Test Transistor Beta, Test Properties Timing

ⓘ The individual options for the timing of the measurement operation are described in more detail in Chapter 5.5.12.1, "Timing", on page 103.

Voltage VBE + Beta	
"Delay"	The fixed delay to the start of the measurement is entered in this field. Value range: 0 ms ... 1 s Resolution: 1 ms Default: 0 ms
"Max. Wait Interval"	The maximum duration of the test interval is entered in this field (maximum 20 samples). Value range: 0 ms ... 10 s Resolution: 1 ms Default: 0 ms

"Max. Wait Accuracy"	In this field, the maximum percentage difference between two measured values in succession (samples) is entered for test steps in which the measured value is applied (measurement with "Max. Wait Interval").
	Value range:
	0.0 % ... 10.0 %
	Default:
	1.0 %
"Average"	The number of measured values from which the mean for the measured result is to be determined is entered in this field.
	Value range:
	1 ... 1000
	Default:
	1
"Sample Interval"	The waiting time between the individual measurements for the formation of the mean is entered in this field (measurement with Average).
	Value range:
	5 s ... 1 s
	Resolution:
	5 µs
	Default:
	5 µs (200 kHz)

5.5.10.6 Results Details

For the Transistor Beta test step, the following information is displayed in the Results/ Details window after the execution of the test step:
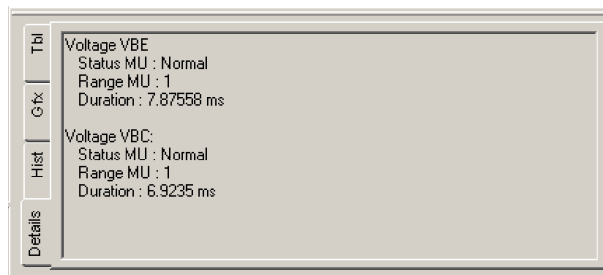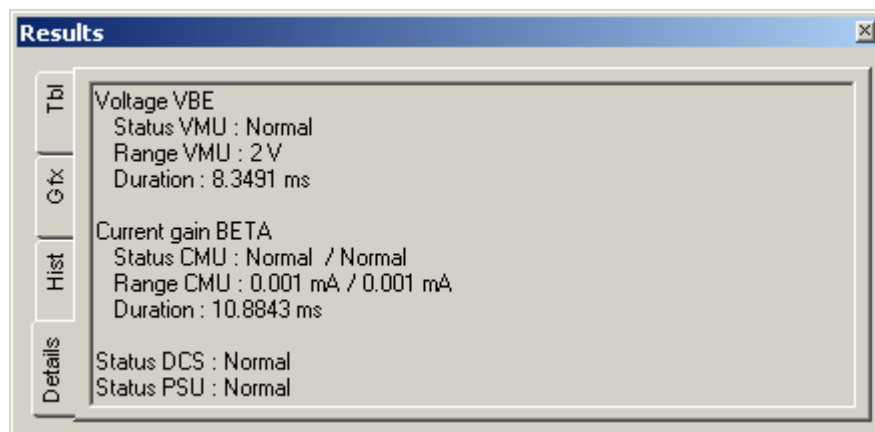

Figure 5-58: Test Step Transistor Beta, Results Details

	VMU = Voltage measuring unit (R&S TS-PSAM)
	CMU = Current measuring unit (R&S TS-PICT)
	DCS = DC Source (R&S TS-PSAM)
	PSU = DC Source (R&S TS-PSU / R&S TS-PSU12)
Voltage VBE	
Status VMU	Status of the voltage measurement unit: Normal, Overrange, Underrange, Max Wait Timeout
Range VMU	Range of the voltage measurement unit, in which the measurement was carried out.
Duration	Duration of the last measurement carried out.
Current gain BETA	
Status CMU	Status of the current measurement unit for the first and second measurement: Normal, Overrange, Underrange, Max Wait Timeout
Range CMU	Range of the current measurement unit, in which the measurement was carried out for the first and second measurement.
Duration	Duration of the last measurement carried out.
Status DCS	Status of the DC voltage source DCS: Normal, Not in constant current state
Status PSU	Status of the DC voltage source TS-PSU / TS-PSU12: Normal, Not in constant voltage state

5.5.11 Zener Diode

ⓘ The Zener Diode test is described in Chapter 10.13, "Zener Diode", on page 215.

5.5.11.1 Limits

Figure 5-59: Test Step Zener Diode, Test Properties Limits

"Nominal"	The nominal value for the zener voltage (reverse working voltage) of the zener diode is entered in this field.
"Upper Limit" "Lower Limit"	The upper and lower limits of the measured zener voltage (reverse working voltage) can be entered in these fields. The fields on the left contain the absolute value in the same unit as the nominal value. The fields on the right contain the deviation from the nominal value as a percentage. Inactive input fields (which appear in gray) are recalculated and displayed with the command **"Edit… Apply"**.
"Absolute / Relative"	This setting determines whether limits will be entered as absolute values or relative values.

5.5.11.2 Settings
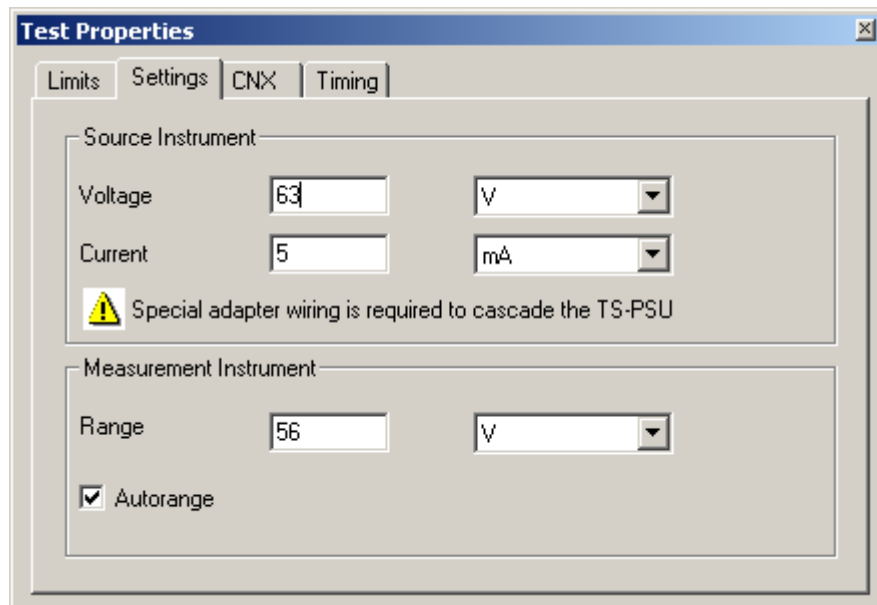
Figure 5-60: Test Step Zener Diode, Test Properties Settings

"Voltage"	The voltage of the voltage source is entered in this field.
	Value range:
	-100 V ... 100 V for TS-PSU
	-24 V ... 24 V for TS-PSU12
	Default:
	3.3 V
"Current"	The current limit of the voltage source is entered in this field.
	Value range:
	1.0 µA … 100.0 mA
	Default:
	5.0 mA

When Voltage exceeds 50 V for TS-PSU (12 V for TS-PSU12) , special test adapter wiring is necessary to cascade the two power supply output channels (see).

"Range"	The measuring range of the voltmeter is entered in this field. Value range: 10.0 mV … 100.0 V Default: 5.0 V
"Autorange"	The measuring range for the voltmeter is set automatically when this check box is selected.

5.5.11.3 CNX
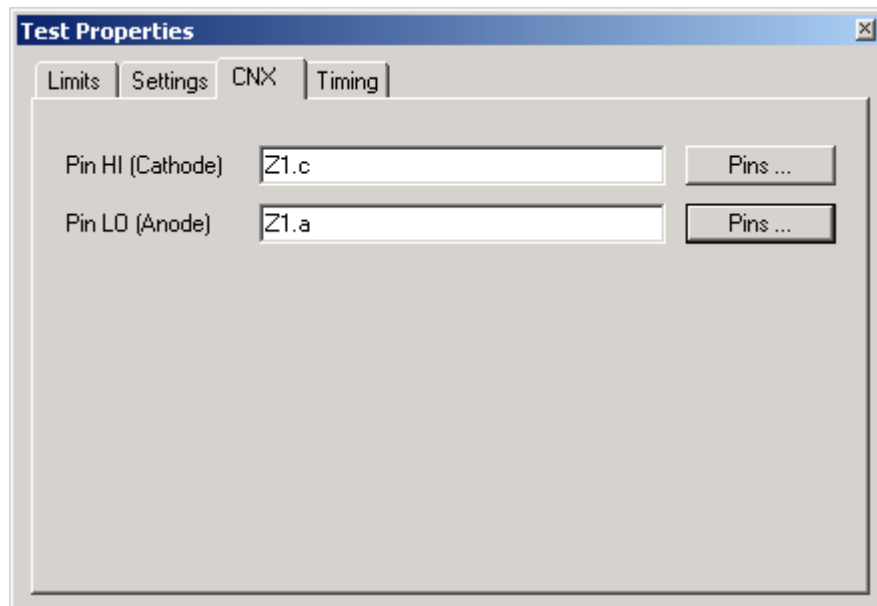
Figure 5-61: Test Step Zener Diode, Test Properties CNX

" Pin HI (Cathode)"	The **Pin HI** (diode cathode) is entered in this field. Only one measuring pin can be entered.
"Pin LO (Anode)"	The **Pin LO** (diode anode) is entered in this field. Only one measuring pin can be entered.
Pins ...	Opens the **Pins** dialog box for inserting / removing pins on the Pin List. For more information, see Chapter 5.5.12.2, "Editing Pin Lists", on page 105.

5.5.11.4 Timing

Figure 5-62: Test Step Zener Diode, Test Properties Timing

ⓘ The individual options for the timing of the measurement operation are described in more detail in Chapter 5.5.12.1, "Timing", on page 103.

"Delay"	The fixed delay to the start of the measurement is entered in this field.
	Value range:
	0 ms ... 1 s
	Resolution:
	1 ms
	Default:
	0 ms
"Max. Wait Interval"	The maximum duration of the test interval is entered in this field (maximum 20 samples).
	Value range:
	0 ms ... 10 s
	Resolution:
	1 ms
	Default:
	0 ms

"Max. Wait Accuracy"	In this field, the maximum percentage difference between two measured values in succession (samples) is entered for test steps in which the measured value is applied (measurement with "Max. Wait Interval").
	Value range:
	0.0 % ... 10.0 %
	Default:
	1.0 %
"Average"	The number of measured values from which the mean for the measured result is to be determined is entered in this field.
	Value range:
	1 ... 1000
	Default:
	1
"Sample Interval"	The waiting time between the individual measurements for the formation of the mean is entered in this field (measurement with Average).
	Value range:
	5 µs ... 1 s
	Resolution:
	5 µs
	Default:
	5 µs (200 kHz)

5.5.11.5 Results Details

For the Zener Diode test step, the following information is displayed in the Results/Details window after the execution of the test step:
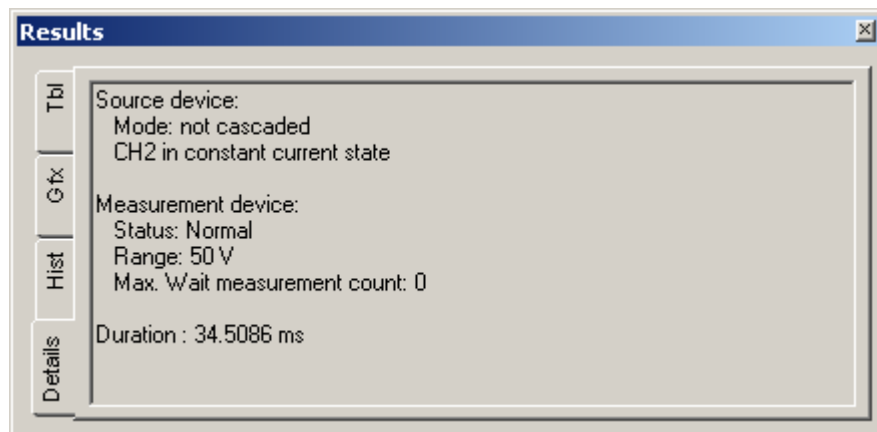
Figure 5-63: Test Step Zener Diode, Results Details

Source device	R&S TS-PSU / R&S TS-PSU12
Mode	Mode of the source output channels: • not cascaded (CH2 only) • cascaded (CH1 and CH2, requires special test adapter cabling)
State	Shows whether the R&S TS-PSU voltage source is in the expected state. The R&S TS-PSU must be in constant current state.
Measurement device	R&S TS-PSAM
Status	Status of the voltage measurement unit: Normal, Overrange, Underrange, Max Wait Timeout
Range	Range of the voltage measurement unit, in which the measurement was carried out.
Max. Wait measurement count	Number of the measurements which were carried out with Max. Wait until the measured value was stable.
Duration	Duration of the last measurement carried out.

5.5.12 Explanations

5.5.12.1 Timing
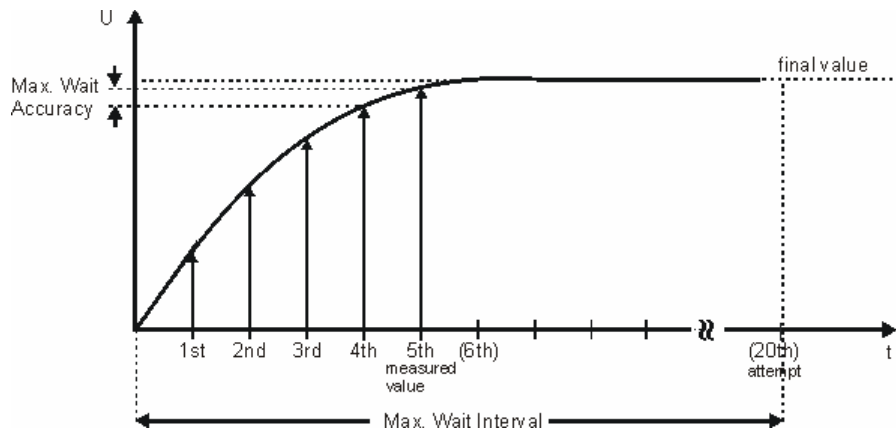
Figure 5-64: Timing example 1
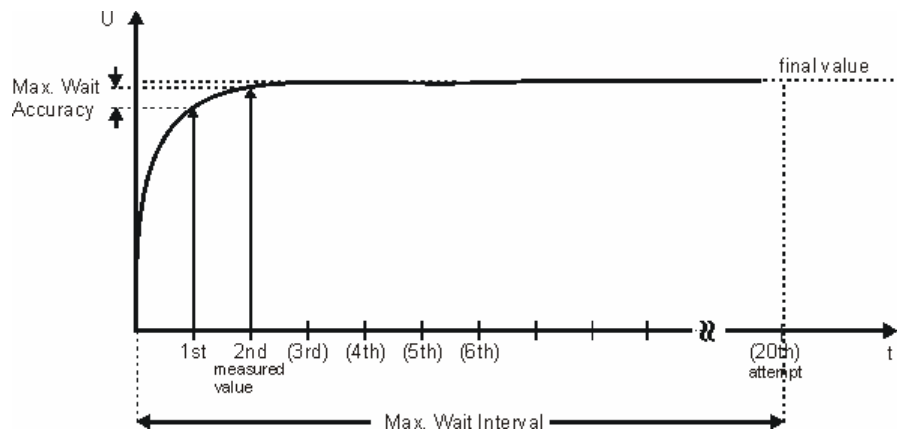
Figure 5-65: Timing example 2
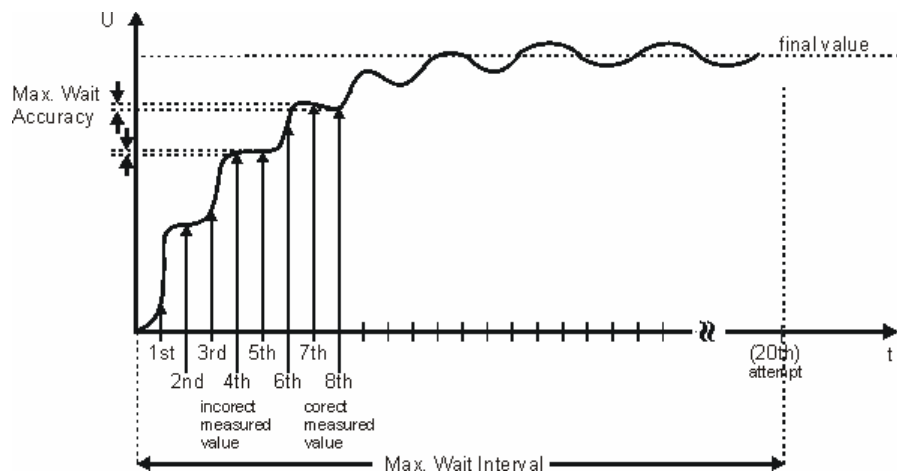
Figure 5-66: Timing example 3

The auto-delay technique (timing) is a technique that automatically adjusts the duration of analog measurements to the unit under test and thus optimizes the measurement duration (see Figure 5-65 to Figure 5-66). There is no additional effort for the programmer. The technique automatically determines the optimal time for the measurement from the transient response of the unit under test. In this way fixed waiting times that must be set unnecessarily long for the worst measuring case are avoided.

During the transient process in Figure 5-64 the relative difference between two individual measurements (**"Max.Wait Accuracy"**) after the 5th measurement is so low that the signal can be considered stable and thus the transient complete. The considerably faster transient signal from Figure 5-65 is already stable after the 2nd measurement so that here only half the measuring time is needed. A shorter **"Max. Wait Interval"** and thus shorter periods between two individual measurements would result in an even shorter measuring time here (e.g. a sampling rate as shown in Figure 5-66).

If the signal to be measured is very unstable, the technique is interrupted after a time that can be programmed by the user (**"Max. Wait Interval"**). During this time 20 measurements are performed and the measured value is checked for stability. A short **"Max. Wait Interval"** thus corresponds to short spacing between the individual measurements, a long time corresponds to a large spacing.

In the realistic case, the signal to be measured will have interference superimposed. The excessively small sampling interval selected in Figure 5-66 already reflects stable conditions after the 4th measurement, although relative stability of the measurement comparable with Figure 5-64 would only be achieved on the 8th measurement.

Half the sampling rate - this corresponds to the even-numbered attempts in Figure 5-66 - would only achieve the value on the 8th measurement. As on real units under test, the response times are scattered by orders of magnitude, the sampling rate must not be constant, instead it must be possible to adapt it to the measuring problem. In this way, the influence of interference is suppressed and high measuring accuracy achieved with optimized measuring time.

(i) If the **"Max. Wait Interval"** is set to 0.0, the auto-delay technique (timing) is switched off.

A further possible method of measuring is to wait for the expected minimum response time using **"Delay"** and then to perform the measurement up to the maximum up to the maximum response time using the auto-delay technique (**"Max. Wait Interval"**) (see Figure 5-67).
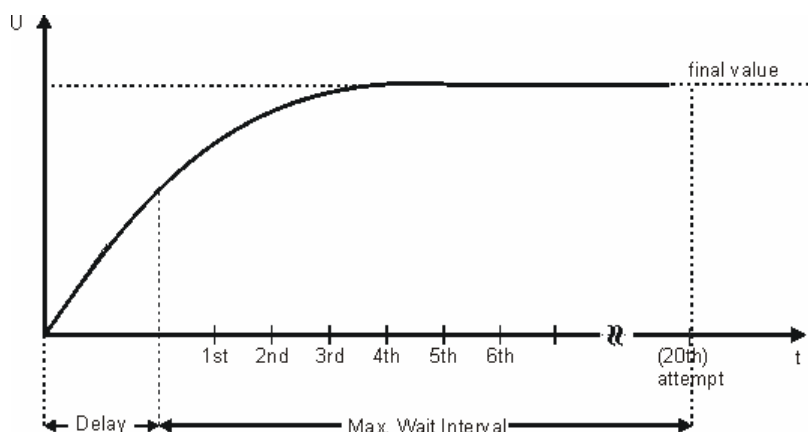
Figure 5-67: Timing example 4

Using **"Sample Interval"** and the formation of the mean (**"Average"**), hum interference can be suppressed. The time set using **"Sample Interval"** is waited prior to each measurement. To suppress 50Hz hum interference, **"Sample Interval"** and **"Average"** must be selected such that the entire measuring time is a multiple of 20 ms. Recommended for the normal case:

"Average" = 20ms, **"Sample Interval"** = 1 ms

5.5.12.2 Editing Pin Lists

Using the **"Pins"** dialog box, the pin lists for the individual test steps are prepared and edited.

Table 5-3: Pins dialog box

Pin Selector	
"Available Pins:"	All available measuring pins are listed in this field. The list of available measuring pins is read from the file "APPLICATION.INI" (on this topic, see also section 7.3).
"Selected Pins:"	All measuring pins that are to be entered in the pin list for the related test step are entered in this field. Depending on the test step, it is possible to copy several pins (Table 5-3, left) or only one pin (Table 5-3, right) to the pin list
>	All marked pins from the **"Available Pins:"** field are entered in the **Selected Pins:** field. Using **Ctrl + left mouse button** or **Shift + left mouse button** you can mark several pins.
>>	All pins from the **"Available Pins:"** field are entered in the **Selected Pins:** field.
<<	All pins from the **"Selected Pins:"** pins field are entered in the **Available Pins:** field.
<	All marked pins from the **"Selected Pins:"** field are entered in the **Available Pins:** field. Using **Ctrl + left mouse button** or **Shift + left mouse button** you can mark several pins.
OK	All pins entered in the **"Selected Pins:"** field are copied to the pin list for the test step. The **Pins** dialog box is closed.
Cancel	All entries made are discarded and the **Pins** dialog box is closed.

5.5.13 User-defined Test Methods

(i) The user-defined test methods are described in Chapter 17, "ICT Extension Libraries", on page 260.

5.5.13.1 Limits

Figure 5-68: Test Step user-defined, Test Properties Limits

"Nominal"	The nominal value for the quantity to be measured is entered in this field.
"Upper Limit" "Lower Limit"	The upper and lower limits of the measured quantity can be entered in these fields. The fields on the left contain the absolute value in the same unit as the nominal value. The fields on the right contain the deviation from the nominal value as a percentage. Inactive input fields (which appear in gray) are recalculated and displayed with the command **"Edit… Apply"**.
"Absolute / Relative"	This setting determines whether limits will be entered as absolute values or relative values.
"Unit"	The unit for the quantity to be measured is entered in this field. See the documentation for the user-defined test about the permissible unit strings.

5.5.13.2   Settings
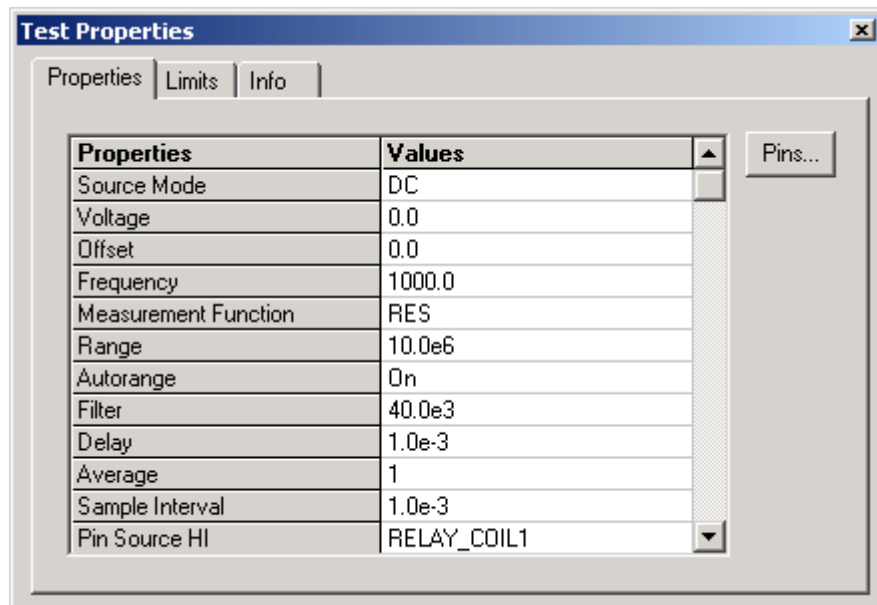
Figure 5-69: Test Step user-defined, Test Properties Settings

The settings for user-defined test methods are shown in the form of a table. The left column **"Properties"** shows the parameters that can be changed. The right column **"Values"** contains the corresponding values. The values in the right column can be edited.

For the meaning of properties, their data types and permissible value ranges, please refer to the documentation for the test method.

Pins...	Opens the Pins dialog box to insert / delete pin names.

5.5.13.3 Info
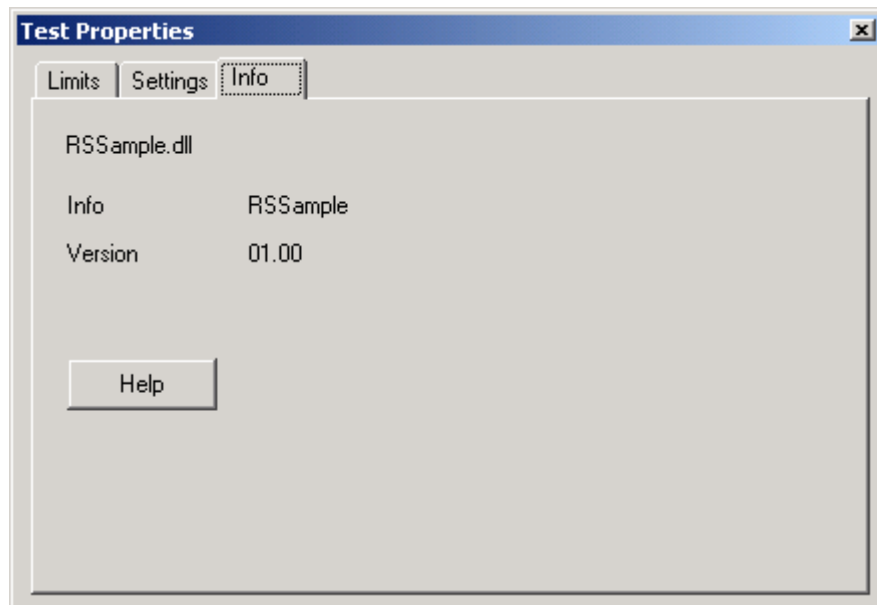
Figure 5-70: Test Step user-defined, Test Properties Info

This property sheet shows the name of the ICT Extension DLL, the name of the user-specific test method and its version number.

Help	Opens the documentation for the user-specific test method.

5.5.13.4 Results Details
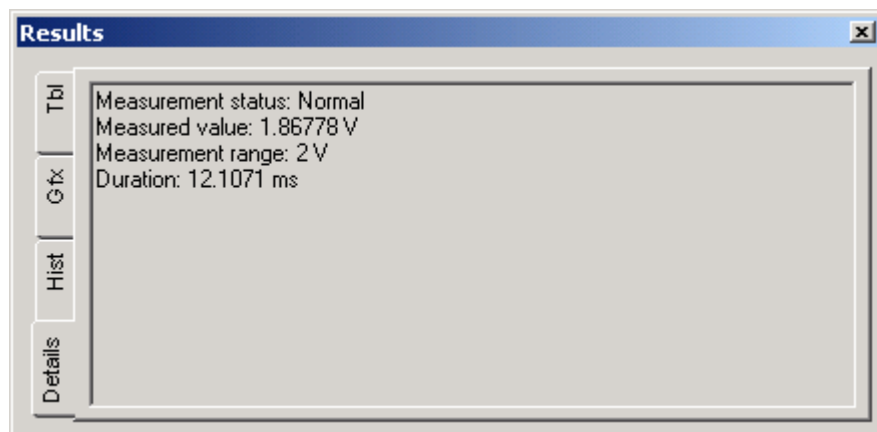
Figure 5-71: Test Step user-defined, Results Details

The data shown here refers to the last test step executed. The meaning of the output is explained in the documentation for the test method.

5.6   Menu bar Functions

5.6.1   Main Menu Command <File>

5.6.1.1   Menu Command <Open>

[Ctrl+O]	Opens the window for opening an ICT program.
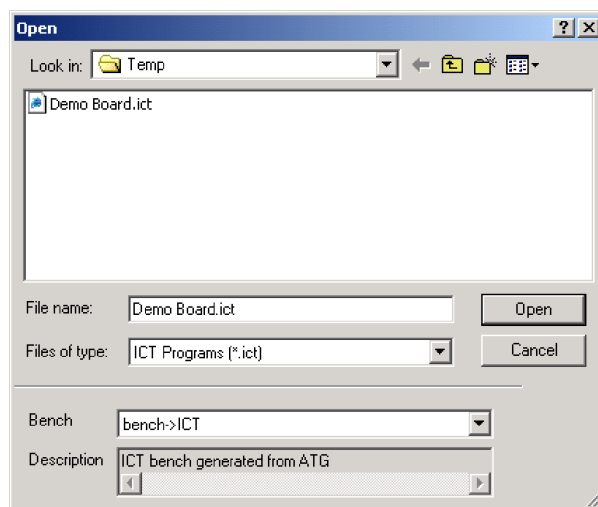
Figure 5-72: Open

The standard Windows dialog box for opening files is displayed. As in every other Windows program, using this dialog box the location, the file name and the file type for the ICT program to be opened can be selected.

"Bench"	In addition, you can enter the bench section for the ICT program using a list box. The bench sections given in the `Application.INI` are listed in the list box.
"Description"	The entry in the selected bench section that is entered under the keyword **Description** is displayed in this field.

When an ICT program is opened, it is "compiled". During compilation, all checks are made. Then a metacode of the ICT program is prepared. When the ICT program is started, only the compiled metacode is executed. As no further check is performed, the execution speed of the ICT program is increased.

5.6.1.2 Menu Command <Save>

[Ctrl+S]	Saves the ICT program displayed using the current name.

5.6.1.3 Menu Command <Save Copy As>

Opens the standard Windows dialog box for saving the ICT program displayed. The ICT program displayed is saved as a copy using a new name. The original ICT program remains open with the old file name.
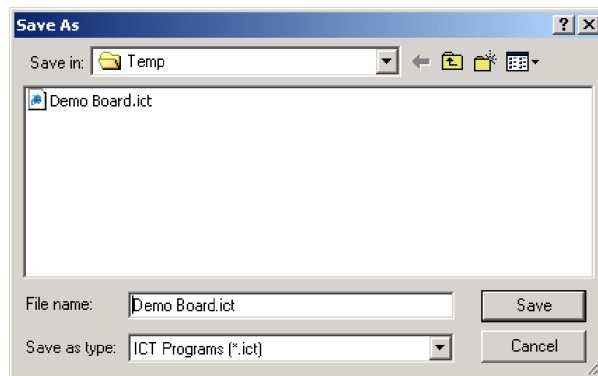
Figure 5-73: Save Copy As

ⓘ Using the Enhanced Generic Test Software Library R&S EGTSL it is not possible to create a new ICT program. A new ICT program is always a copy of an existing ICT program. During the installation of R&S EGTSL an ICT program is supplied as a template for copying.

```
...GTSL\EGTSL\Templates\Template.ict
```

5.6.1.4 Menu Command <Select>

In the R&S EGTSL display environment, you can open several ICT programs. However, only one ICT program can be displayed and edited. Using the **"Select"** menu command, you can select the program to be edited from the ICT programs open.
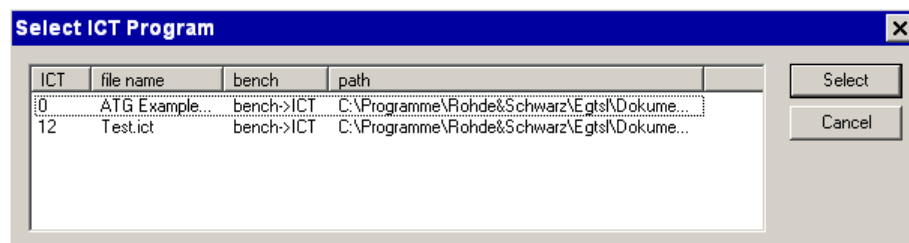
Figure 5-74: Select ICT Program

| Select | The ICT program marked is displayed and can be edited. |
| Cancel | The new selection for the ICT program is discarded and the window closed. |

5.6.1.5 Menu Command <Close>

The ICT program displayed is closed. If changes have been made to the ICT program and not yet saved, a warning message is displayed.

5.6.1.6 Menu Command <Program Properties>

Opens a window where you can make general settings for the ICT program. The following buttons have the same function in all subwindows of the Program Properties.

| OK | All changes made are applied and saved. The window is closed. |
| Cancel | All changes made are discarded. The window is closed. |
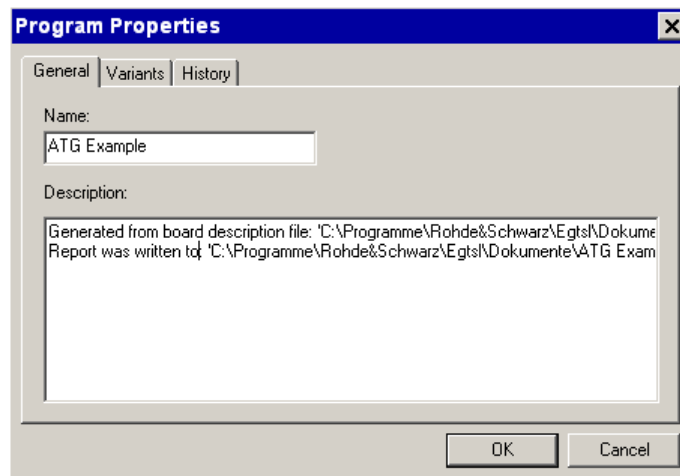
Figure 5-75: ICT Program Properties, General

In the *Program Properties, General* window, general information on the ICT program is given.

| "Name": | The name of the ICT program is entered in this field. The name appears as the root folder name in the tree structure in the Program window (see Chapter 5.4.2, "Program Subwindow", on page 40). |
| "Description": | Arbitrary text describing the ICT program can be entered in this window. Text can be copied to the window using the Clipboard. |
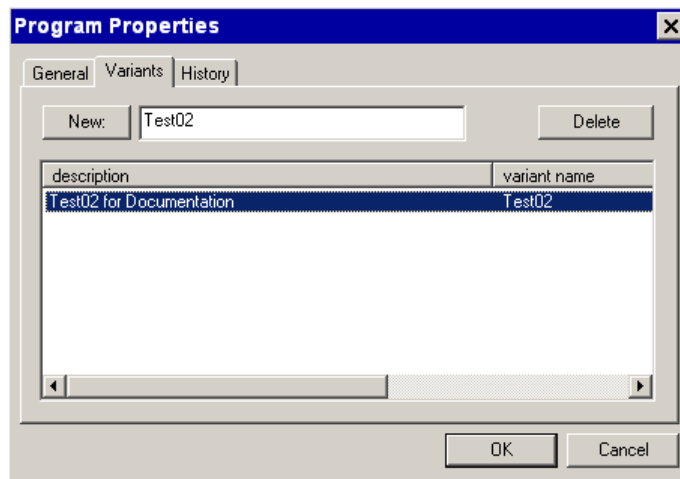
Figure 5-76: ICT Program Properties, Variants

In the *Program Properties, Variants* window you can define the variants that are to apply for the ICT program.

ⓘ For more information on variants, see Chapter 11.2, "Variants", on page 219.

New:	The variant name entered in the field is copied to the list using the **"New"** button.
Delete	Deletes the marked variant entry from the list. Once a variant is allocated to a test step or a program group, this variant cannot be deleted. The **"Delete"** button is hidden.
"description"	The description of the variant is entered in this column. The text can be edited by clicking the entry marked.
"variant name"	The variant name entered is copied to this column. The variant name cannot be changed later.
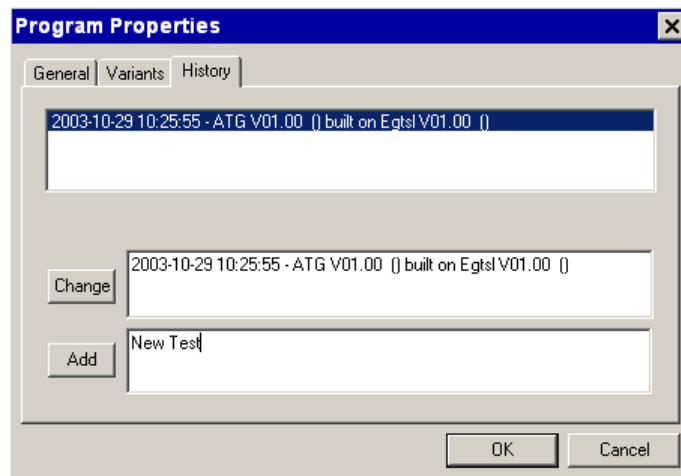
Figure 5-77: ICT Program Properties, History

In the *Program Properties, History* window the history of the ICT program is logged. The information must be entered manually.

A list of all entries is displayed in the window at the top.

Change	The entry marked in the window is changed. Using the **"Change"** button the change is copied to the list.
Add	A new entry is made in the window. Using the **"Add"** button the new entry is entered at the top of the list.

5.6.1.7 Menu Command <Limits>

Opens the editing window for loading, importing and exporting the values for the limits for the individual test steps.

Figure 5-78: Limits

Source/Destination	
"Clipboard"	With this function activated, the data for loading, importing and exporting values for limits are written to the Clipboard or read from the Clipboard.
"File"	With this function activated, the data for loading, importing and exporting values for limits are written to a `txt` file or read from a `txt` file.
Browse..	Opens the standard Windows window for selecting an existing file with values for limits. In the standard Windows window, it is also possible to create a file for exporting the values for the limits. The folder and the file name for the selected export/import file are displayed next to the **"Browse..."** button and next to the **"View File"** button.
View File	Opens the selected export/import file in the Windows text editor. Changes can be made and saved in the editor.
"Format:"	There is a choice of two data formats for exporting or importing: • Tab-delimited Text (file extension `*.txt`) The individual entries on a line are separated by tabs (see example 1) • Comma-delimited Text (file extension `*.csv`) The individual entries on a line are separated by commas (see example 2) The data formats apply • for Importing (loading) from the Clipboard. • Exporting to the Clipboard. • Importing (loading) from a file. Exporting to a file.
"Start Of Data Marker:"	The start of the data area in the import/export file is marked with the text given here (format setting).
"End Of Data Marker:"	The end of the data area in the import/export file is marked with the text given here (format setting).
"Skip Rows That Begin With:"	Lines in the import/export file that are to be skipped during loading/importing are marked with the characters (text) given here (format setting).
Load	Loads the values for the limits from the Clipboard or the import file given. The values for the limits are temporarily changed and displayed. The values loaded cannot be saved. When the ICT program is opened next time, the original values for the limits will be displayed again. The data format given and the format settings must match the information in the file to be loaded.
Import	Imports the values for the limits from the Clipboard or the import file given. The old values for the limits are overwritten and saved with the ICT program. The data format given and the format settings must match the information in the file to be imported.

Export	Exports the values for the limits for all test steps (when present) to the Clipboard or the export file given. The data format from the format settings is applied during the export
Close	The editing window for loading, importing and exporting the values for the limits is closed.

Examples for an export/import file with values for limits:

Example:

Start ICT Limits				
<Step Name>	Limits.High	Limits.Low	[Limits.High	Limits.Low]
// example of a comment line				
Discharge Test	0	0		
Contact Test	0	0		
Short Test	0	0		
R1	1.3538	1.0462		
R2	365.58	294.45		
R3	44.135	35.868		
R4	11.035	8.9674		
C1	396.42	323.68		
C2	110.16	89.94		
C3	110.1	89.896		
TR1	0.78	0.42	0.78	0.42
End ICT Limits				

Example:
```
START SECTION LIMITS
<Step Name>,Limits.High,Limits.Low,[Limits.High,Limits.Low]
// example of a comment line
Discharge Test,0,0
Contact Test,0,0
Short Test,0,0
R1,1.3538,1.0462
R2,365.58,294.45
R3,44.135,35.868
R4,11.035,8.9674
C1,396.42,323.68
C2,110.16,89.94
C3,110.1,89.896
TR1,0.78,0.42,0.78,0.42
END SECTION LIMITS
```

The following points are to be observed when editing the export/import file with values for the limits:

- The numerical values have the English number format i.e. the decimal point and not the decimal comma must be used.
- The names of the individual tests are given in the first column. If there are test steps with the same test step name (e.g. the resistor test Rx twice), the test step names are expanded by the test step ID .

Example:

```
Rx[27]
Rx[28]
```

- Resistor and impedance tests have no second limit. The column "Limits.High" for the second limit is used to store the optional correction value.

5.6.1.8 Menu Command <Print>

[Ctrl+P]	Opens the standard Windows print dialog box for printing the report.

It is not possible to print an ICT program. It is only possible to print the contents of the Report window
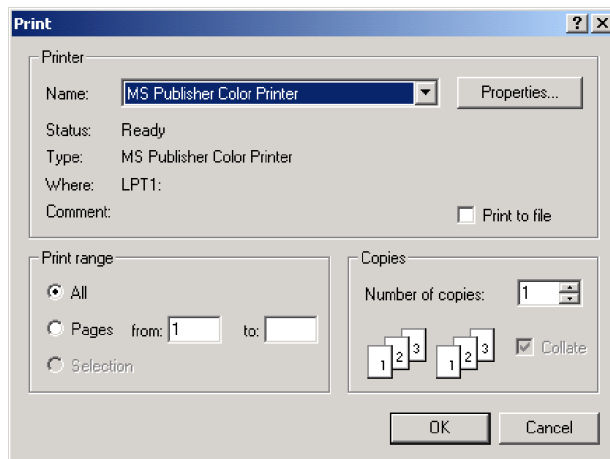
Figure 5-79: Print

All marked areas of the Report window are printed. The entire report can be marked using the context menu (see).

5.6.1.9 Menu Command <Print Setup>

Opens the standard Windows printer dialog box with the settings for the printer connected.
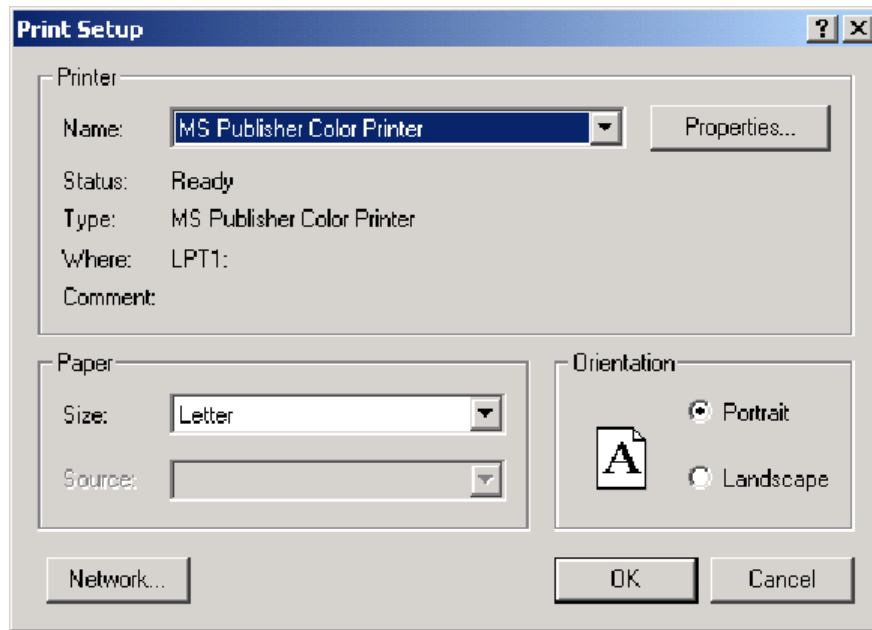
Figure 5-80: Print Setup

ⓘ The standard Windows functions are used.

5.6.1.10 Menu Command <Exit>

Quits the program Enhanced Generic Test Software Library R&S EGTSL. If changes have been made to the ICT program opened and these have not yet been saved, a warning message is displayed (see Figure 5-81).
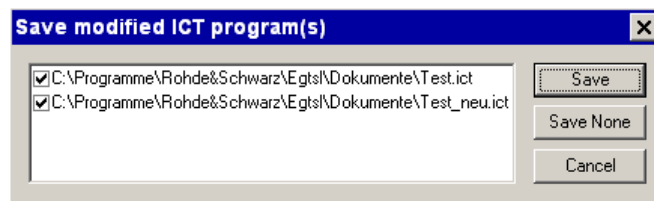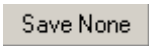
Figure 5-81: Exit Warning

The ICT programs that have not yet been saved are listed.

Save	The ICT programs marked with a check mark are saved. The changes to ICT programs that are not marked are discarded. The program is quit.
Save None	None of the ICT programs displayed is saved. All changes made are discarded. The program is quit.
Cancel	The quitting of the program is canceled. The ICT program last edited is displayed.

5.6.2 Main Menu Command <Edit>

> The descriptions of the menu commands that only execute functions and for which a dedicated window is not opened are summarized in Chapter 5.6.2.1, "Functions", on page 119.

5.6.2.1 Functions

Undo [Ctrl+Z]	Undoes the last action. Up to 256 actions can be undone.
Redo [Ctrl+Y]	Reverses the action of the <Undo> command.
Cut [Ctrl+X]	Removes the item currently marked (test step, program group, text content) and places it in the Clipboard.
Copy [Ctrl+C]	Copies the item currently marked (test step, program group, text content) to the Clipboard.
Paste [Ctrl+V]	Inserts the contents of the Clipboard (test step, program group, text content) at the current cursor position.
Delete [Ctrl+Del]	Deletes a marked test step or the marked program group.
Insert	Opens a list box for inserting test steps. On this topic, see also section 5.5: Test Steps.
Select All [Ctrl+A]	Marks all test steps or all entries in the Report window. The function depends on the cursor position.
Apply [Ctrl+Enter]	The changes made in the test properties for the test step marked are applied. The entries are checked for plausibility and compliance with the value ranges. The metacode for the ICT program is updated.
Revert [Ctrl+Back]	The changes made in the test properties for the test step marked are discarded. The values that applied when the test step was marked are displayed again.
Auto apply	When this button is activated, the changes made in the test properties for the test step marked are automatically applied when you change to a different test step. The entries are checked for plausibility and compliance with the value ranges. The metacode for the ICT program is updated.
Rename [F2]	Enables the name of the marked test step to be edited.

5.6.2.2   Menu Command <Find>

[Ctrl+F] 🔭	Opens the dialog box for the selection of the search criteria for individual test step names or program group names (see Figure 5-83) or for text in the Report window (see Figure 5-84). The function depends on the cursor position.

Search for text step name or program group name:
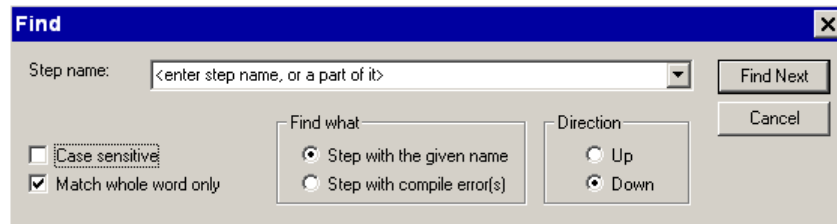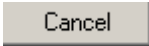
Figure 5-82: Find (Program)

"Step name"	The search text for the test step name or program group name is entered in this field. The most recent search texts are saved and can be selected again using the list box. When R&S EGTSL is quit, the saved search texts are deleted.
"Case sensitive"	When this function is activated, the search is only performed for test step names or program group names with the same upper and lower case as in the **Step name** field.
"Match whole word only"	With this function activated, the search is only performed for test step names or program group names that exactly match the information in the **Step name** field.
Find what	
"Step with the given name"	With this function activated, the search is performed for test step names or program group names that match the information in the **Step name** field.
"Step with compile error(s)"	With this function activated, the search is only performed for test step names that have a compile error. The information in the **"Step name"** field is ignored.
Direction	
"Up"	Defines the direction starting from the cursor position in which the ICT
"Down"	program is to be searched. With the **Up** function activated, the search is performed from the cursor position upwards. With the **Down** function activated, the search is performed from the cursor position downwards.

| Find Next | The search is continued to the next entry (test step name, program group name) that matches the entry made. The test step or the program group found is marked. |
| Cancel | The search in the Program window is stopped. |

Searching for text in the report:
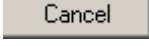
Figure 5-83: Find (Report)

"Find what"	The search text is entered in this field.
"Match case"	With this function activated, the search is only per-formed for text with the same upper and lower case as in the **"Find what"** field.
Direction	
"Up"	Defines the direction starting from the cursor posi-tion in which the report is to be searched. With the **Up** function activated, the search is performed from the cursor position upwards. With the **Down** function activated, the search is performed from the cursor position downwards.
"Down"	
Find Next	The search is continued for the next entry that matches the information given. The test entry found is marked.
Cancel	The test search in the Report window is stopped.

5.6.2.3 Menu Command <Step Properties>

| [Alt+Enter] | Opens the dialog button for the step properties for the test step or program group marked. |

The dialog box for the step properties can also be opened using the program context menu (see Chapter 5.4.2.1, "Context Menu", on page 41).

The *Step Properties, Common* (see Figure 5-84) are displayed when opened from a marked test step or program group.
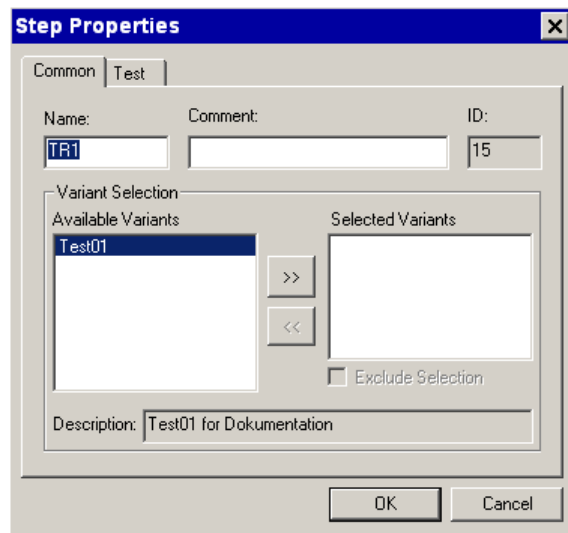
Figure 5-84: Step Properties, Common

"Name"	The name of the test step or program group is edited in this field.
"Comment"	The comment for the test step or for the program group is entered in this field. The text entered is displayed in the Program window in the **Comment** column.
"ID"	To be able to identify uniquely each test step and each program group in the program, each entry is automatically allocated an ID number when it is created. This **ID** is displayed in this window.
Variant Selection	
"Available Variants"	All variants for the ICT program are listed in this list. Variants are added in the program properties (see Chapter 5.6.1.6, "Menu Command <Program Properties>", on page 112).
Selected Variants	All variants allocated to the test step or the program group are given in this list. Several variants can be allocated to a test step or a program group. The test step or the program group is only executed when the corresponding variant has been selected in the Debug subwindow (see Chapter 5.4.6, "Debug Subwindow", on page 46).
Exclude Selection	With this function activated, the test step or the program group is executed for all variants and the <default> setting. The test step or the program group is not executed with the variants that are listed in the **Selected Variants** list.
>>	Moves the marked variant entry from the **Available Variants** list to the **Selected Variants** list.
<<	Moves the marked variant entry from the **Selected Variants** list to the **Available Variants** list.

Description	A short description of the variants is displayed in this field (see Chapter 5.6.1.6, "Menu Command <Program Properties>", on page 112).
OK	All changes made are applied and saved. The dialog box is closed.
Cancel	All changes made are discarded. The dialog box is closed.

ⓘ For more information on variants, see Chapter 11.2, "Variants", on page 219.

The *Step Properties, Test* (see Figure 5-85) are only displayed when opened from a marked test step.
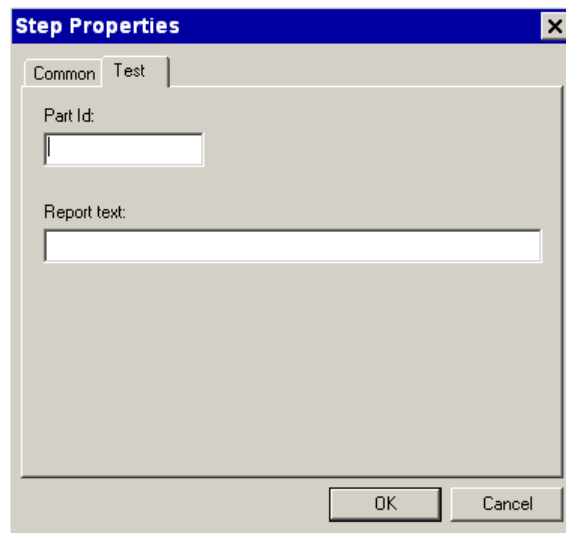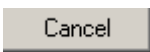
Figure 5-85: Step Properties, Test

"Part Id"	The type identifier or the ident number of the component to be tested is entered in this field. The entry is displayed in the Report window.
"Report text"	Error instructions or repair instructions for the component to be tested are entered in this field. The entry is only displayed in the Report window if there is an error.
OK	All changes made are applied and saved. The dialog box is closed.
Cancel	All changes made are discarded. The dialog box is closed.

The *Step Properties, Group* (see Figure 5-86) are only displayed when opened from a marked program group.
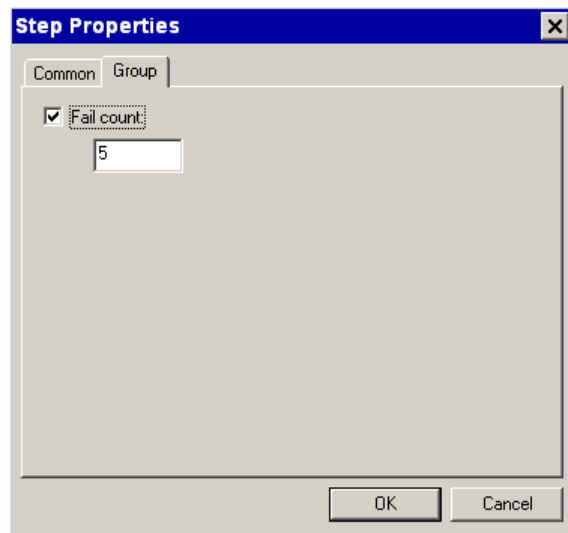
Figure 5-86: Step Properties, Group

" Fail count"	With this function activated, the erroneous test steps in a program group are counted. If the number of erroneous test steps given is reached, none of test steps that follow in the program group are executed.
OK	All changes made are applied and saved. The dialog box is closed.
Cancel	All changes made are discarded. The dialog box is closed.

5.6.2.4 Menu Point <Breakpoints>

[Alt+F9]	Opens the dialog box for editing breakpoints.
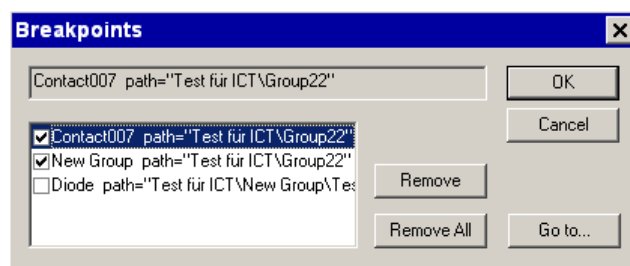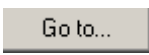
Figure 5-87: Breakpoints

A list with all the breakpoints added to the ICT program is displayed. The test step name or the program group name and the folder for the breakpoint added are displayed. Breakpoints marked with a check mark are active. Breakpoints that are not marked with a check mark are deactivated. The entry for the marked breakpoint is displayed in the top line.

Remove	The marked breakpoint is deleted from the list and the ICT program.
Remove All	All breakpoints are deleted from the list and the ICT program.
Go to...	A jump is made to the marked breakpoint in the ICT program. The dialog box is closed. The corresponding test step or the corresponding program group is marked.
OK	All changes made are applied and saved. The dialog box is closed.
Cancel	All changes made are discarded. The dialog box is closed.

5.6.3   Main Menu Command <View>

" Main Toolbar"	This command hides and displays the Main toolbar.
"Debug Toolbar"	This command hides and displays the Debug toolbar.
"Toolbar for Insert"	This command hides and displays the Insert toolbar.
"User-defined Toolbar"	This command hides and displays the toolbar for user-defined test steps.
"Status Bar"	This command hides and displays the Status bar at the bottom of the screen.
"Results" *[Alt+0]*	This command hides and displays the Result window.
"Test Properties" *[Alt+1]*	This command hides and displays the Test Properties window.
"Debug" *[Alt+2]*	This command hides and displays the Debug window.

5.6.4   Main Menu Command <Debug>

	Text in status bar
"Debugger activated"	Debugger ready
"Debugger deactivated"	Debugger deactivated

"Go" *[F5]*	Executes the ICT program once from the test step marked with the yellow arrow (current debugger execution point) to the end. If the debugger is deactivated, the ICT program is executed from start to finish.
"Restart" *[Ctrl+Shift+F5]*	With the debugger activated, executes the ICT program from the start to the finish.
"Terminate" *[Shift+F5]*	Stops the execution of the ICT program. The marking with the yellow arrow (current debugger execution point) is removed. The debugger is deactivated.
"Break"	Interrupts the execution of the ICT program. The test step at which the interruption takes place is marked with the yellow arrow (current debugger execution point) and displayed. Using the **Go** command the execution of the ICT program can be continued.
"Repeat" *[F4]*	With the debugger activated, starts the execution of the test steps marked (test steps with a color background) with the number of repetitions given.
"Resume" *[Shift+F4]*	Continues the execution of the marked test steps (test steps with a color background) following an interruption using the **Break** button.
"Step Into " *[F11]*	Executes the ICT program from the test step marked with the yellow arrow (current execution point of the debugger) in single steps. If the test step is a group, automatically each test step of the group will be executed in single steps.
"Step Over" *[F10]*	Executes the ICT program from the test step marked with the yellow arrow (current execution point of the debugger) in single steps. If the test step is a group, all test steps in the group are performed automatically.
"Step Out" *[Shift+F11]*	If the test step marked with the yellow arrow (current execution point of the debugger) is in a group, the group is left with this button. The test steps that follow in the group are performed automatically. If the test step marked with the yellow arrow (current execution point of the debugger) is at the top program level, all following test steps including groups are executed.
"Run to Cursor" *[Ctrl+F10]*	Executes the ICT program from the test step marked with the yellow arrow (current execution point of the debugger) to the test step marked with the cursor. The test step marked by the position of the cursor is marked as the next test step that can be executed using the yellow arrow.
"Set Next Step" *[Ctrl+Shift+F10]*	Marks the position of the cursor as the test step that is to be executed next. A yellow arrow is displayed on the left beside the test step marked. The debugger is activated.

"Show Next Step " *[Alt+F10]*	Jumps to the current debugger execution position (test step marked with the yellow arrow). The test step is the next that is executed.
"Toggle Breakpoint" *[F9]*	Sets a breakpoint at the cursor position. The breakpoint is deactivated when you click the button again. The third time you click the button, the breakpoint is deleted (see also Chapter 5.6.2.4, "Menu Point <Breakpoints>", on page 124).
"Stop At Fail "	With this function activated, the ICT program execution is stopped when the test step result was "Fail". The program stops at the erroneous test step. The **Stop at fail** function applies to all debug commands with the exception of the **Repeat** function.
"Break At Fail"	With this function activated, the execution of the ICT program (test steps marked) is interrupted when the test step result was "Fail". The program stops at the erroneous test step. The **Break at fail** function applies only to the **Repeat** function.

5.6.5 Main Menu Command <Report>

" Clear Report Window"	Deletes all entries in the Report window.
"Report Enable"	With this function activated a report is displayed in the Report window.
"Report Errors Only"	With this function activated, only erroneous measurements and error messages are displayed in the Report window.
"Clear Results"	Deletes all entries in the Result windows (Tbl, Gfx, Hist, Details) for all test steps.
"Auto Clear"	With this function activated, all entries in the Result window (Tbl, Gfx, Hist, Details) will be deleted for all test steps on any test step command (e.g. Run, Step, Repeat).

5.6.6 Main Menu Command <Help>

5.6.6.1 Menu Command <About EGTSL>

	Opens the window with information on the software.
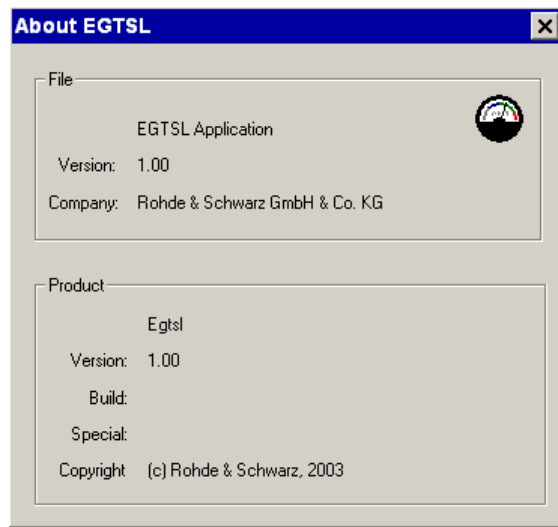
Figure 5-88: About EGTSL

For further inquiries and technical support about the Enhanced Generic Test Software Library R&S EGTSL the information from this window are required.

5.7   Toolbar Functions

5.7.1   Main Toolbar

	Open Opens the window for opening an ICT program. See **<Open>** menu command in Chapter 5.6.1.1, "Menu Command <Open>", on page 110.
	Save Saves the ICT program using the current name. See **<Save>** menu command in Chapter 5.6.1.2, "Menu Command <Save>", on page 111.
	Print Opens the standard printer dialog box for printing the report. See **<Print>** menu command in Chapter 5.6.1.8, "Menu Command <Print>", on page 117.
	Cut Removes the item currently marked (test step, text content, program group) and places it in the Clipboard.
	Copy Copies the item currently marked (test step, text content, program group) to the Clipboard.

	Paste Inserts the contents of the Clipboard (test step, text content, program group) at the current position of the cursor.
	Undo Undoes the last action.
	Redo Reverses the action of the **<Undo>** command.
	Auto Apply When this button is activated, the changes made in the test properties for the test step marked are applied automatically when you change test step. The entries are checked for plausibility and compliance with the value ranges. The metacode for the ICT program is updated.
	Apply The changes made in the test properties for the test step marked are applied. The entries are checked for plausibility and compliance with the value ranges. The metacode for the ICT program is updated.
	Revert The changes made in the test properties for the test step marked are discarded. The values that applied when the test step was marked are displayed again.
	Step Properties Opens the dialog box for the entry of the general step properties. See <**Step Properties**> menu command in Chapter 5.6.2.3, "Menu Command <Step Properties>", on page 121.
	Find Opens the dialog box for selecting the search criteria for individual test steps or for text content in the Report window. The function depends on the position of the cursor. See **<Find>** menu command in Chapter 5.6.2.2, "Menu Command <Find>", on page 120.
	About Open the window with information on the software. See <**About EGTSL**> menu command in Chapter 5.6.6.1, "Menu Command <About EGTSL>", on page 127.

5.7.2 Debug Toolbar

	Go Executes the ICT program once from the test step marked with the yellow arrow (current debugger execution point) to the end. If the debugger is deactivated, the ICT program is executed from start to finish.
	Restart With the debugger activated executes the ICT program from the start to the finish.
	Terminate Stops the execution of the ICT program. The marking with the yellow arrow (current debugger execution point) is removed. The debugger is deactivated.
	Break Interrupts the execution of the ICT program. The test step at which the interruption takes place is marked with the yellow arrow (current debugger execution point) and displayed. Using the **Go** command the execution of the ICT program can be continued.
	Step Into Executes the ICT program from the test step marked with the yellow arrow (current execution point of the debugger) in single steps. If the test step is a group, automatically each test step of the group will be executed in single steps.
	Step Over Executes the ICT program from the test step marked with the yellow arrow (current execution point of the debugger) in single steps. If the test step is a group, all test steps in the group are performed automatically.
	Step Out If the test step marked with the yellow arrow (current execution point of the debugger) is in a group, the group is left with this button. The test steps that follow in the group are performed automatically. If the test step marked with the yellow arrow (current execution point of the debugger) is at the top program level, all following test steps including groups are executed.
	Run to Cursor Executes the ICT program from the test step marked with the yellow arrow (current execution point of the debugger) to the test step marked with the cursor. The test step marked by the position of the cursor is marked as the next test step that can be executed using the yellow arrow.

	Set Next Step
	Marks the position of the cursor as the test step that is to be executed next. A yellow arrow is displayed on the left beside the test step marked. The debugger is activated.
	Show Next Step
	Jumps to the current debugger execution position (test step marked with the yellow arrow). The test step is the next that is executed.
	Toggle Breakpoint
	Sets a breakpoint at the cursor position. The breakpoint is deactivated when you click the button again. The third time you click the button, the breakpoint is deleted (see also Chapter 5.6.2.4, "Menu Point <Breakpoints>", on page 124).
	Stop At Fail
	With this function activated, the ICT program execution is stopped when the test step result was "Fail". The program stops at the erroneous test step. The **Stop at fail** function applies to all debug commands except for the **Repeat** function.
	Clear report window
	Deletes all entries in the Report window.
	Clear Results
	Deletes all entries in the Result windows (Tbl, Gfx, Hist, Details) for all test steps.

5.7.3 Toolbar for Insert

On this topic, see also Chapter 5.5, "Test Steps", on page 47.

	Insert Group
	Inserts a new group above the cursor position.
	Insert Contact
	Inserts a contact test step above the cursor position.
	Insert Continuity
	Inserts a continuity test step above the cursor position.
	Insert Diode
	Inserts a diode test step above the cursor.
	Insert Discharge
	Inserts a discharge test step above the cursor.
	Insert FET
	Inserts a FET test step above the cursor.

	Insert Impedance
	Inserts an impedance test step above the cursor position.
	Insert Resistor
	Inserts a resistor test step above the cursor position.
	Insert Short
	Inserts a short-circuit test step above the cursor position.
	Insert Transistor
	Inserts a transistor test step above the cursor position.
	Insert Transistor Beta
	Inserts a transistor beta test step above the cursor position.
	Insert Zener Diode
	Inserts a zener diode test step above the cursor.

5.7.4 Toolbar for User-defined Tests

On this topic, see also Chapter 5.5.13, "User-defined Test Methods", on page 107.

	Insert <user-defined test>
	Inserts a user-defined test step above the cursor position. (The icon displayed is specific to the user-defined test extension library.)

5.8 Shortcuts

SHORTCUT	Command
Ctrl+O	**Open**
Ctrl+S	**Save**
Ctrl+P	**Print**
Ctrl+Z	**Undo**
Ctrl+Y	**Redo**
Ctrl+X	**Cut**
Ctrl+C	**Copy**
Ctrl+V	**Paste**
Ctrl+Del	**Delete**
Ctrl+A	**Select All**

SHORTCUT	Command
Ctrl+Enter	**Apply**
Ctrl+Back	**Revert**
Ctrl+F	**Find**
Alt+Enter	**Step Properties**
Alt+0	Hides and displays the Results subwindow.
Alt+1	Hides and displays the Test Properties subwindow.
Alt+2	Hides and displays the Debug subwindow.
Alt+F10	**Show Next Step**

FUNCTION KEY	Command
F2	**Rename**
F3	Find next
F4	**Repeat**
Shift+F4	**Resume**
F5	**Go**
Shift+F5	**Terminate**
Ctrl+Shift+F5	**Restart**
F6	Next subwindow
Shift+F6	Previous subwindow
F9	**Toggle Breakpoint**
Alt+F9	**Breakpoints**
F10	**Step Over**
Ctrl+F10	**Run to Cursor**
Shift+F10	Context menu
Ctrl+Shift+F10	**Set Next Step**
F11	**Step Into**
Shift+F11	**Step Out**

6 License Management

The Enhanced Generic Test Software Library R&S EGTSL is part of the Generic Test Software Library R&S GTSL.

ⓘ Starting with GTSL 3.30, no GTSL license is required.

During the installation of the Generic Test Software Library R&S GTSL, all available test libraries and R&S EGTSL are copied to the system. You need a License Key File in order to access the functions from the test libraries and R&S EGTSL.

Refer to chapter 7 of the "Software Description Generic Test Software Library R&S GTSL" for the license key required for each test library and R&S EGTSL.

Without a valid License Key File, the functions of R&S EGTSL will only work in "demo mode". Access to the hardware is only simulated.
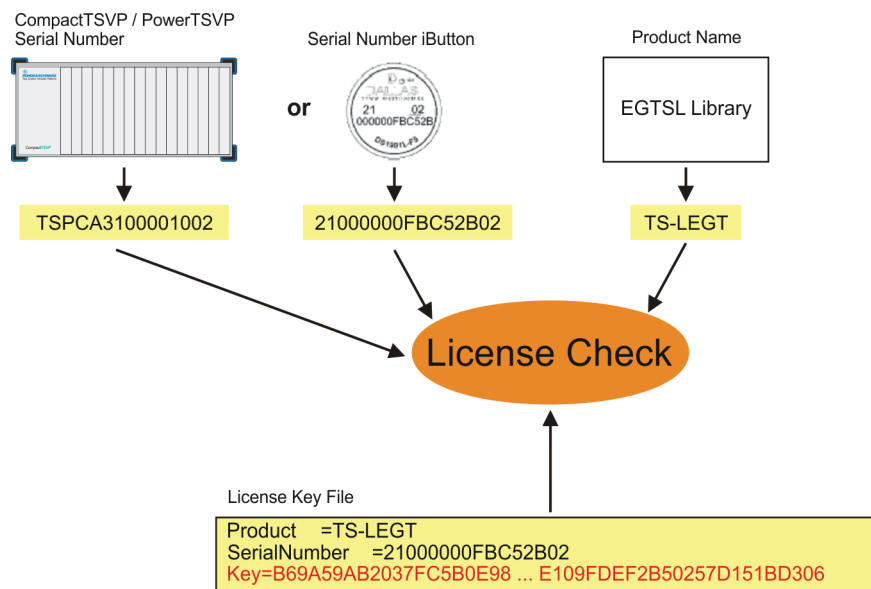
Figure 6-1: License checking

Each license is bound to a system serial number. The enabled R&S EGTSL can only be run on the system with this serial number. The hardware system used is identified via the system module PSYS (R&S CompactTSVP, R&S PowerTSVP) or via an iButton (TSVP, desktop PC, notebook). The iButton is located on a Hardlock-Adapter, which is plugged into the serial interface (usually COM1) of the system controller of the Test System Versatile Platform TSVP or of the connected control computer. The system module or iButton has a unique serial number whereby the hardware system can be unambiguously identified.

During license checking, the serial number and the name of the called test library (R&S EGTSL) are compared with the License Key from the License Key File. The test

library (R&S EGTSL) in question will only be enabled if these coincide. The serial number and the name of the test library (R&S EGTSL) are encoded in the License Key.

Example of a License Key File:

```
[Header]
FileVer=1.0
[Project]
Info=EGTSL
[Modul]
Product=TS-LEGT
SerialNumber=850000008E4BD202
Key=C146648E1DEF9AD78663728A5D8E8D25885F457367D7F7C359F2C63BDB926 ...
```

The serial number is queried and a new License Key File is installed via the R&S GTSL License Viewer. To open the R&S GTSL License Viewer, select the following:

"Start -> Programs -> GTSL -> License Viewer"

The R&S GTSL License Viewer is described in section 4 of the "Software Description Generic Test Software Library R&S GTSL".

7 Configuration Files

The required configuration files are stored by default in the

`..\GTSL\Configuration` directory.

7.1 Syntax

The syntax of the Physical Layer Configuration File (`PHYSICAL.INI`) and of the Application Layer Configuration File (`APPLICATION.INI`) is identical. The only difference between the two files is in terms of how they are used (see sections Chapter 7.2, "PHYSICAL.INI", on page 140 and Chapter 7.3, "APPLICATION.INI", on page 143).

Both files use the standard INI file format.

Example:

Standard INI file format:

```
[section]

key = value

...
```

A section begins with the section name written inside closed brackets ([ ]). The following lines contain pairs of keywords and values. The keywords and the assigned values are separated by an equals sign ("=").

In the section names and keywords, no distinction is made between upper and lower case characters. However, the values after the equals sign are transferred exactly as they are written in the file. Leading and trailing spaces are truncated.

7.1.1 Naming Conventions

In the Physical Layer Configuration File and in the Application Layer Configuration File, several groups of keywords and sections are allowed. These refer to other sections and reflect the relationships and interconnections. The section names follow the naming conventions indicated below.

The section name begins with the section type followed by an arrow ("->", a minus sign followed by a greater than sign). A unique name appears after the arrow. No spaces are permitted between the name and the arrow.

In the section names, no distinction is made between upper and lower case characters.

The following characters are permitted for the logical names, the device names and the bench names.

Table 7-1: Character set for names

"A" ... "Z"	Upper case characters
"a" ... "z"	Lower case characters
"0" ... "9"	Numbers
"_"	Underscore
"."	Decimal point

The following maximum character lengths are permitted for section names, keywords and values:

Table 7-2: Maximum character lengths

section	80 characters
key	80 characters
value	260 characters

[LogicalNames]	This section contains a list of names of devices and benches. Any name can be used to identify a device or bench (Application Layer Configuration File).
[Device->...]	A device section contains different keywords to identify the devices. These include the GPIB address, the device type etc. (Physical Layer Configuration File and Application Layer Configuration File).
[Bench->...]	This section contains a group of device entries which together form a bench. A High Level Library requires the name of a bench in its setup routine (Application Layer Configuration File).
[ResourceManager]	This section contains information for the configuration of the Resource Manager.

7.1.2 [LogicalNames] Section

The [LogicalNames] section is used to assign a short, meaningful name to a device or bench. Any name can be chosen. This section contains a list of unique name allocations. The values on the right side of the expressions must be valid names of a bench or of a device section.

The [LogicalNames] section is an optional entry and is used only in the Application Layer Configuration File.

Example:

```
[LogicalNames]
ICT = bench->ICT
SIMU = bench->ICT_Simulation
```

7.1.3 [Device] Section

The [Device] section contains a list of keywords and assigned values. These keywords and values precisely describe the relevant device. The name of the [Device] section begins with "Device->" followed by a unique name. Any name can be chosen.

There must be a [Device] section for each device in the Physical Layer Configuration File.

A [Device] section with the same name can be defined in the Application Layer Configuration File. Additional device information can be given at this point by means of further keywords and values, or device information from the Physical Layer Configuration File can be overwritten. However, it is not possible to define a [Device] section in the Application Layer Configuration File which is not present in the Physical Layer Configuration File.

The keywords in a [Device] section and their meaning depend on the libraries used by the devices.

Table 7-3: Standard keywords of [Device] section

Keyword	Description
Description	*Optional entry* Device description, remarks
Type	*Mandatory entry* Device type (e.g. PSAM, PICT, PMB etc.)
ResourceDesc	*Mandatory entry* VISA device properties and device description in the form: GPIB[card number]::[primary address]::[secondary address] PXI[segment number]::[device number]::[function]::INSTR CAN[board]::[controller]::[frame]::[slot] Example: "CAN0::0::1::15"
DriverSetup	*Optional entry* Special setup string for IVI driver, e.g. for simulation of devices

The "Type" and "ResourceDesc" entries are required for the test libraries (e.g. ICT.DLL). Both entries must be present in the Physical Layer Configuration File.

The information from the "Type" entry allows the test libraries to distinguish between different supported devices (such as PMB or PSM1). This information is also needed for the system self-test. The information from the "ResourceDesc" entry is needed to set up the device driver and create the physical connection with the indicated device.

Example:

```
[device->pmb1]
Description  = "TS-PMB, Matrix Module, Slot 15"
Type        = PMB
ResourceDesc = CAN0::0::1::15
```

7.1.4   [Bench] Section

The [Bench] section contains a list of keywords and assigned values which describes a group of devices and their use. The name of the [Bench] section begins with "Bench->" followed by a unique name. Any name can be chosen. A [Bench] section can only be defined in the Application Layer Configuration File.

The keywords in a [Bench] section depend on the test library used by the bench. A keyword always provides at least one reference to a device entry. Other keywords may be necessary to describe the bench. The following keywords are predefined and should be present in each [Bench] section.

Table 7-4: Standard keywords of [Bench] section

Keyword	Description
Description	*Optional entry* Bench description, remarks
Simulation	*Optional entry* If set to "1", the complete bench is simulated by the test library.
Trace	*Optional entry* If set to "1", the tracing function is enabled for the test library

The [Bench] section can contain further useful keywords and values which are used by a test library. For the test library GSM.LIB, entries for the calibration, the calibration files, the network etc. may be useful.

Example:

```
[bench->ICT]
Description     = ICT bench (Simulation)
Simulation      = 1
Trace           = 0
ICTDevice1      = device->psam
ICTDevice2      = device->pict
SwitchDevice1   = device->pmb1
AppChannelTable = io_channel->ICT
```

7.1.5   [ResourceManager] Section

The [ResourceManager] section contains keywords and assigned values to control the behavior of the Resource Manager library. The following keywords are supported:

Table 7-5: Keywords of [ResourceManager] section

Key name	Remarks
Trace	Blocks the tracing function (value = 0), enables the tracing function (value = 1). The function impacts on all libraries.
TraceFile	Defines the path and the name of the trace file.
TraceToScreen	The tracing information is displayed on the standard screen (value = 1).
TraceTimeStamp	Writes the time of day at the start of each tracing line (value = 1).
TraceThreadID	Writes the ID of the current thread at the start of each tracing line (value = 1).

7.1.6   [ExtIct] Section

The [ExtIct] section contains keywords and assigned values for the ICT extension libraries. The following keywords are supported:

Table 7-6: Keywords of [ExtIct] section

Key name	Remarks
UserDefinedDll	Defines the path and name of the ICT extension DLL.

7.2   PHYSICAL.INI

In the file `PHYSICAL.INI` (Physical Layer Configuration File), all hardware assemblies available in the Enhanced Generic Test Software Library R&S EGTSL are described along with the corresponding definitions and settings (see example `PHYSICAL.INI` file). This file also contains definitions which are applicable to all test applications to be executed on the system (e.g. type definition). The information entered in this file is used by all test libraries and thus by each test step.

The `PHYSICAL.INI` file normally exists only once in the system as it reflects the exact physical structure. The file must only be modified in the event of a hardware change.

The Resource Manager calls and administers the information from the `PHYSICAL.INI` file.

The file `PHYSICAL.INI` is saved in the folder `...\GTSL\Configuration\physical.ini`

7.2.1 Example File for PHYSICAL.INI (Example_Physical.ini)

	Description
`[device->psam]`	1
`Description = "TS-PSAM, Source and Measurement Module, Slot 8"`	2
`Type = PSAM`	3
`ResourceDesc = PXI1::10::0::INSTR`	4
`DriverDll = rspsam.dll`	5
`DriverPrefix = rspsam`	6
`DriverOption = "Simulate=0,RangeCheck=1"`	7
`[device->pict]`	1
`Description = "TS-PICT, ICT Extension Module, Slot 9"`	2
`Type = PICT`	3
`ResourceDesc = PXI2::15::0::INSTR`	4
`DriverDll = rspict.dll`	5
`DriverPrefix = rspict`	6
`DriverOption = "Simulate=0,RangeCheck=1"`	7
`SFTDll = sftmpict.dll`	8
`SFTPrefix = SFTMPICT`	9
`[device->pmb1]`	1
`Description = "TS-PMB, Matrix Module, Slot 15"`	2
`Type = PMB`	3
`ResourceDesc = CAN0::0::1::15`	4
`DriverDll = rspmb.dll`	5
`DriverPrefix = rspmb`	6
`DriverOption = "Simulate=0,RangeCheck=1"`	7
`SFTDll = sftmpmb.dll`	8
`SFTPrefix = SFTMPMB`	9
`; Analog Bus`	10
`[device->ABUS]`	1
`Description = "Analog Bus"`	2
`Type = AB`	3

7.2.2 Description of example file PHYSICAL.INI

The description is based on the example file in Chapter 7.2.1, "Example File for PHYS-ICAL.INI (Example_Physical.ini)", on page 141. The indicated numbers refer to the corresponding positions in the example file. The place-holder "XY" in the following listing stands for the corresponding entries.

Table 7-7: Description of PHYSICAL.INI

1	[device->XY]	Defines the name under which the device is called in the test libraries. A separate entry must be made for each device. The entry in square brackets [ ] defines a new section within which new definitions are made.
2	Description = "XY"	Gives a detailed description of the defined device. The entry is optional.
3	Type = XY	Gives the exact designation of the defined device. This designation is needed to call the corresponding device driver. The entry is mandatory.
4	ResourceDesc = XY	Gives the necessary hardware information required for the defined device. The entry is mandatory. Details provided at this point include, for example: GPIB address: GPIB1::20::1 (example) GPIB[card number]::[primary address]::[secondary address] Serial interface: COMX PXI address: PXI1::10::0::INSTR (example) PXI[segment number]::[device number]::[function]::INSTR CAN-address: CAN0::0::1::5 (example) CAN[board]::[controller]::[frame]::[slot]
5	DriverDll = XY	Gives the path and the file name of the device driver.
6	DriverPrefix = XY	Gives a prefix for the device driver.
7	Driver Option = XY	Gives certain options applicable to the device driver.
8	SFTDll = XY	Gives the path and the file name of the selftest device driver.
9	SFTPrefix = XY	Gives a prefix for the selftest device driver.
10		Text appearing after a semicolon (;) is interpreted as a comment.

7.3 APPLICATION.INI

In the `APPLICATION.INI` file (Application Layer Configuration File) is a description of how the individual test libraries and the test functions use the hardware components (see example file `APPLICATION.INI`). Different hardware components can be combined into groups (bench). This bench can then be used within the test function. Furthermore, definitions are made in this file which apply to certain test applications to be executed on the system (e.g. definition of designations in the case of multi-channel operation).

The Resource Manager calls and administers the information from the `APPLICATION.INI` file.

ⓘ A special Application Layer Configuration File is produced by the Automatic Test Generator ATG for each ICT program generated. The name of the Application Layer Configuration File is given in the ATG.

During the execution of the ICT program, the corresponding Application Layer Configuration File must be given.

7.3.1 Example File for APPLICATION.INI (Example_Application.ini)

	Description
`[ResourceManager]`	1
`; general trace settings (normally off)`	2
`Trace = 0`	3
`TraceFile = %GTSLROOT%\resmgr_trace.txt`	4
`;-------------------------------------------------------`	2
`[ExtIct]`	5
`UserDefinedDll =` `%GTSLROOT%\EGTSL\ExtIct\RSSample\RSSample.dll`	6
`UserDefinedDll = C:\UserDefinedDlls\MyOpAmp.dll`	6
`UserDefinedDll = C:\UserDefinedDlls\MyRelay.dll`	6
`;-------------------------------------------------------`	2
`[LogicalNames]`	7
`ICT = bench->ICT`	8
`;-------------------------------------------------------`	2

	Description
`[bench->ICT]`	9
`Description = ICT bench (Simulation)`	10
`Simulation = 1`	11
`Trace = 0`	12
`ICTDevice1 = device->PSAM`	13
`ICTDevice2 = device->PICT`	14
`SwitchDevice1 = device->PMB1`	15
`AppChannelTable = io_channel->ICT`	16
`AppWiringTable = io_wiring->ICT`	17
`[io_channel->ICT]`	18
`GND = PMB1!P1`	19
`INPUT = PMB1!P2`	19
`OUTPUT = PMB1!P3`	19
`TR1.B = PMB1!P4`	19
`TR1.C = PMB1!P5`	19
`TR1.E = PMB1!P6`	19
`VCC = PMB1!P7`	19
`;------------------------------------------------------------`	2
`[io_wiring->ICT]`	20
`GND = F1 S15 X10A1`	21
`INPUT = F1 S15 X10A2`	21
`OUTPUT = F1 S15 X10A3`	21
`TR1.B = F1 S15 X10A4`	21
`TR1.C = F1 S15 X10A5`	21
`TR1.E = F1 S15 X10A6`	21
`VCC = F1 S15 X10A7`	21
`;------------------------------------------------------------`	

7.3.2 Description of Example File APPLICATION.INI

The description is based on the example file in Chapter 7.3.1, "Example File for APPLI-CATION.INI (Example_Application.ini)", on page 143. The indicated numbers refer to the corresponding positions in the example file. The place-holder "XY" in the following listing stands for the corresponding entries.

Table 7-8: Description of APPLICATION.INI

1		[ResourceManager]	Defines a new section (identified by the square brackets [ ]) with information evaluated directly by the Resource Manager.
2			Text appearing after a semicolon (;) is interpreted as a comment.
3		Trace = 0 Trace = 1	Tracing of information is not carried out (Trace = 0) or is carried out (Trace = 1).
4		TraceFile = XY	Gives the path and file name for storing trace information.
5 + 6		[ExtIct] UserDefinedDll = XY	Defines a section in which the user-defined ICT extension DLLs are listed.
7 to 8		[LogicalNames]	Defines a new section in which logical short names are defined. The short names can be used to call the libraries.
9		[bench->XY]	Defines a new bench with its name. The name, which is defined at this point, is called in the SETUP routine of the corresponding test library.
10		Description = "XY"	Provides a detailed description of the bench defined. The entry is optional.
11		Simulation = 0 Simulation = 1	Defines whether the execution is only to be simulated (Simulation = 1) or specific measurements are to be performed (Simulation = 0).
12		Trace = 0 Trace = 1	Tracing of information is not carried out (Trace = 0) or is carried out (Trace = 1).
13*		ICTDevice1 = XY	Refers to the entry for the R&S TS-PSAM. The addressed devices must be defined in the PHYSICAL.INI file with their details.
14		ICTDevice2 = XY	Refers (optional) to the entry for the R&S TS-PICT. Without the R&S TS-PICT some test methods cannot be used, e.g. impedance test. The addressed devices must be defined in the PHYSICAL.INI file with their details.

15*	SwitchDevice1 = XY	Refers to the entry for the R&S TS-PMB. As an option, it is possible to refer to further R&S TS-PMB modules using the entries SwitchDevice2 ... n. The addressed devices must be defined in the PHYSICAL.INI file with their details.
16*	AppChannelTable = XY	Refers to the section with the definitions of the I/O channels that are to apply for this bench.
17	AppWiringTable = XY	Refers to the section with the definitions of the wiring table that is to apply for this bench.
18* + 19*	[io_channel->XY]	The definitions of I/O channels that follow apply for the applications that are to be run with the bench given. The test points (nodes) that have been determined from the circuit description (BDL file) by the Automatic Test Generator ATG are on the left. The I/O channels on the R&S TS-PMB Matrix Card allocated by the ATG are on the right.
20 + 21	[io_wiring->XY]	The physical wiring from the UUT pins to the front connectors of the R&S TS-PMB Matrix Cards. The test points (nodes) are on the left. The front connector locations are on the right in the form: F<frame number> S<slot number> X10<column><row> F1 S15 X10A1 means: - TSVP frame 1 - Slot 15 - Front connector X10, column A, row 1.

For an in-circuit tests always the items marked with an asterisk (*) must exist in the `APPLICATION.INI` file.

8 Circuit Description

8.1 Definition of Meta Language

In the following a meta language is explained that is used for the description of the syntax for the external representation of data. This metalanguage is a derivative of the Backus-Naur form.

8.1.1 Terminology, Symbols

The text that contains the formal description of the syntax comprises a sequence of so-called "tokens". A token is either an identifier, a literal or an operator.

Identifier

An identifier is a string of lower case letters, numbers and special character. Each identifier must start with a lower case letter that can then be followed by any number of lower case letters, numbers or special characters.

Example:

ict-compilation

program-header-stmt

execution

dd

Literals

Literals comprise a chain of alphanumeric characters that are enclosed in quotation marks "". If there are also quotation marks in the literal, each quotation mark must be given twice.

Example:

"voltage"

"Literal with quotation mark "" in the text"

"SYSTEM"

Operators

The operators are used to be able to better represent the structure of a metaprogram. The following operators are used:

:	=	Separator between rule name and rule body
%	=	The preceding token occurs 1 to n-times
*	=	The preceding token occurs 0 to n-times

n	=	The preceding token occurs exactly n-times
n-m	=	The preceding token occurs at least n-times, however m-times at the most
|	=	Alternative
{	=	Start of the use of parenthesis for several tokens for a unit
{	=	End of the use of parenthesis for several tokens for a unit
[	=	Start of the optional use of parenthesis for one or more tokens
]	=	End of the optional use of parenthesis for one or more tokens
;	=	Conclusion of a production rule
!	=	Character for comment

Layout

For separation of the individual tokens as many spaces as required can be used.

Comments

Using the operator '!' a comment is started that is intended to make it easier to read the metaprogram. The comment starts after the operator and concludes at the end of the line.

8.1.2 Structure of a Metaprogram

The formal description of a language syntax is called a 'metaprogram'. A metaprogram comprises a sequence of production rules.

A production rule comprises the following elements:

- The rule name that is used for referencing the production rule in other production rules
- The operator ':' for separating rule names and rule body
- The rule body that defines the syntax or part of the syntax and
- a semicolon that concludes the production rule.

A metaprogram is complete when

- Each rule name only occurs once and describes exactly one production rule
- There is exactly one and only one rule name that is not in a rule body. This rule forms the start of the syntax.

8.1.3 Structure of Rule Bodies

Every rule body contains a sequence of identifiers or literals that are separated or linked by operators.

Daisy Chaining

If a rule is to comprise several tokens one after the other, then these tokens are simply written one after the other, separated by at least 1 space.

Example:

while : "WHILE" condition "DO" ;

Information on Alternatives

If two or more tokens are to be allowed alternately in a rule, then this is represented by the '|' operator.

Example:

If the rule a is to be able to comprise the rule body b, as well as alternatively the rule body c, the alternative operator is used:

a: b | c ;

Optional Information

If a token is only to be present in a rule as an option, i.e. it can be present, but can also be missing, then this token must be placed in parenthesis using the "[" and "]" operators.

Example:

a: b; or a: bc;

Can be represented by

a: b[c];

Multiple Repetition

If a token is to be repeated several times in a rule, one of the following repetition operators can be given after the token:

"%"   Means that the preceding token must occur at least once, but can also occur as any number of times.

Example:

The following rules are equivalent:

a: b[n]; and a: b%;

"*"   Means that the preceding token does not need to occur at all, and that it can also occur any number of times.

Example:

The following rules are equivalent:

a: [b[n]]; and a: b*;

"n"   Means that the preceding token must occur exactly n-times.

Example:

The following rules are equivalent:

a: b[cccc]; and a: b[c4];

as well as

a: bcdcdcd ; and a: b{cd}3;

"n-m"   Means that the preceding token must occur at least n-times, but is allowed to occur a maximum of m-times.

Example:

The following rules are equivalent:

a: bcc[c][c][c]; and a: bc2-5;

Use of Parentheses

If several tokens are to be combined logically to form a single, complex token, this can be achieved by placing the tokens in parenthesis using the "{" and "}" operators.

Example:

With the aid of the rule a: bcd{efg}*; the following information is allowed syntactically:

bcd

bcdefg

bcdefgefg

bcdefgefgefg

Binding of the operators

The individual operators in the above defined metalanguage have the following priorities:

strongest

[]

{}

% * n n-m

|

weakest
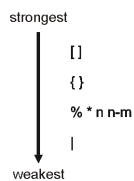
Figure 8-1: Priorities of the operators

8.2 External File Format (BDL)

8.2.1 Syntax

The syntax of the external circuit description is described in a metalanguage that is defined in Chapter 8.1, "Definition of Meta Language", on page 147.

board_description:

```
{   resistor_list
  | var_res_list
  | poti_list
  | res_array_list
  | cap_list
  | pol_cap_list
  | inductor_list
  | diode_list
  | led_list
  | z_diode_list
  | transistor_list
  | jumper_list
  | ic_list
  | connector_list
  | black_box_list
  | track_list
  | node_list
}*  ;
```

8.2.1.1 Resistor List

resistor_list

```
    "RESISTOR"
    { res_element }*   ;


res_element :
    element_name
    {         { part_id }
      |    {      "VALUE"          uns_real         ! Must be present
                   ["K"   |         "M"] ["OHM"] }
      |    {      "TOL+"           uns_real ["%"] }  ! Default 10
      |    {      "TOL-"           uns_real ["%"] }  ! Default 10
      |    {      "PIN_1"          node_name }
      |    {      "PIN_2"          node_name }
      |    {      "I_LIM"          uns_real ["MA"] } ! Default 100
      |    { err_msg }
    }*  ;                  ! Each parameter only once
```

8.2.1.2 Variable Resistor List

var_res_list:

```
    "VAR_RES"
    { var_res_element }*   ;


var_res_element :
    element_name
    {          { part_id }
        |    {     "VALUE"        uns_real          ! Must be present
                   ["K"   |       "M"] ["OHM"] }
        |    {     "TOL+"         uns_real ["%"] }  ! Default 10
        |    {     "TOL-"         uns_real ["%"] }  ! Default 10
        |    {     "SETUP"        uns_real
                   ["K"   |       "M"] ["OHM"]
                   ["ADJ"  |      "FIX"] }          ! Default FIX
        |    {     "PIN_1"        node_name }
        |    {     "PIN_2"        node_name }
        |    {     "I_LIM"        uns_real ["MA"] } ! Default 100
        |    { err_msg }
    }*   ;                    ! Each parameter only once
```

8.2.1.3 Potentiometer List

poti_list:

```
    "POTI"
    { poti_element }*   ;


poti_element :
    element_name
    {          { part_id }
        |    {     "VALUE"        uns_real          ! Must be present
                   ["K"   |       "M"] ["OHM"] }
        |    {     "TOL+"         uns_real ["%"] }  ! Default 10
        |    {     "TOL-"         uns_real ["%"] }  ! Default 10
        |    {     "R_2_3"        uns_real
                   ["K"   |       "M"] ["OHM"]
                   ["ADJ"|        "FIX"] }          ! Default FIX
        |    {     "PIN_1"        node_name }
        |    {     "PIN_2"        node_name }
        |    {     "PIN_3"        node_name }
        |    {     "I_LIM"        uns_real ["MA"] } ! Default 100
        |    { err_msg }
    }*   ;                    ! Each parameter only once
```

8.2.1.4   Resistor Array List

res_array_list:

```
"RESISTOR_ARRAY"
{ res_array_element }*  ;


res_array_element :
    element_name
    {          {      part_id }
        |    {      "ARRAY"        uns_integer              ! Must be present
                    "X"            uns_real                 ! Resistor value
                                                           ! Must be present
                   ["K" |         "M"] ["OHM"] }
        |    {      "TOL+"         uns_real ["%"] }         ! Default 10
        |    {      "TOL-"         uns_real ["%"] }         ! Default 10
        |    {      "I_LIM"        uns_real ["MA"] }        ! Default 100
        |    {      "PIN_"         uns_integer_node_name  }*
        |    { err_msg }
    };                        ! Each parameter only once
```

8.2.1.5   Capacitor List

cap_list:

```
"CAPACITOR"
{ cap_element }*   ;


cap_element :
    element_name
    {          {      part_id }
        |    {      "VALUE"        uns_real        ! Must be present
                    [  "M"
                    |"U"
                    | "N"
                    | "P"
                    ]
                    ["F"] }
        |    {      "TOL+"         uns_real ["%"] }  ! Default 10
        |    {      "TOL-"         uns_real ["%"] }  ! Default 10
        |    {      "PIN_1"        node_name }
        |    {      "PIN_2"        node_name }
        |    {      "FREQ"         uns_real ["KHZ"] }
        |    { err_msg }
    }*  ;                ! Each parameter only once
```

8.2.1.6   Pol. Capacitor List

pol_cap_list:

```
"POL_CAP"
{ pol_cap_element }*   ;


pol_cap_element :
    element_name
    {        {    part_id }
    |    {    "VALUE"        uns_real        ! Must be present
                            [  "M"
                            |"U"
                            |"N"
                            |"P"
                            ]
                            ["F"] }
    |    {    "TOL+"         uns_real ["%"] }  ! Default 10
    |    {    "TOL-"         uns_real ["%"] }  ! Default 10
    |    {    "PIN_A"        node_name }
    |    {    "PIN_C"        node_name }
    |    {    "FREQ"         uns_real ["KHZ"] }
    |    {    err_msg }
    }*   ;               ! Each parameter only once
```

8.2.1.7   Inductor List

inductor_list:

```
"INDUCTOR"
{ ind_element }*   ;


ind_element :
    element_name
    {        {    part_id }
    |    {    "VALUE"        uns_real          ! Must be present
                            ["M"] ["H"]
    |    {    "TOL+"         uns_real ["%"] }      ! Default 20
    |    {    "TOL-"         uns_real ["%"] }      ! Default 20
    |    {    "PIN_1"        node_name }
    |    {    "PIN_2"        node_name }
    |    {    "DC_RES"       uns_real  } ["OHM"] } ! Default 1,0
    |    {    "R_TOL+"       uns_real ["%"] }      ! Default 30
    |    {    "R_TOL-"       uns_real ["%"] }      ! Default 30
    |    {    "FREQ"         uns_real ["KHZ"] }
    |    {    err_msg }
    }*   ;                          ! Each parameter only once
```

8.2.1.8   Diode List

diode_list

```
"DIODE"
{ diode_element }*   ;
```

```
diode_element :
    element_name
    {           {       part_id }
        |       {       "VALUE"          uns_real ["V"] }              ! Default 0.7
        |       {       "TOL+"           uns_real ["%"] }              ! Default 25
        |       {       "TOL-"           uns_real ["%"] }              ! Default 25
        |       {       "PIN_A"          node_name }
        |       {       "PIN_C"          node_name }
        |       {       "I_DCS"          uns_real  } ["MA"] }          ! Default 100
        |       {       "I_DIO"          uns_real  } ["MA"] }          ! Default 10
        |       {       err_msg }
    }*  ;                       ! Each parameter only once
```

8.2.1.9 LED List

led_list:

```
"LED"
{ led_element }*   ;


led_element :
    element_name
    {           {       part_id }
        |       {       "VALUE"          uns_real ["V"] }              ! Default 1.2
        |       {       "TOL+"           uns_real ["%"] }              ! Default 25
        |       {       "TOL-"           uns_real ["%"] }              ! Default 25
        |       {       "PIN_A"          node_name }
        |       {       "PIN_C"          node_name }
        |       {       "I_DCS"          uns_real  } ["MA"] }          ! Default 100
        |       {       "I_DIO"          uns_real  } ["MA"] }          ! Default 10
        |       {       err_msg }
    }*  ;                     ! Each parameter only once
```

8.2.1.10 Zener Diode List

z_diode_list:

```
"Z_DIODE"
{ z_diode_element }*   ;


z_diode_element :
    element_name
    {           {       part_id }
        |       {       "VALUE"          uns_real ["V"] }              ! Must be present
        |       {       "TOL+"           uns_real ["%"] }              ! Default 10
        |       {       "TOL-"           uns_real ["%"] }              ! Default 10
        |       {       "PIN_A"          node_name }
        |       {       "PIN_C"          node_name }
```

```
                  |     {     "I_Z"          uns_real  } ["MA"] }
                  |     {     "R_Z"          uns_real  } ["OHM"] }
                  |     {     err_msg }
         }*  ;                    ! Each parameter only once
```

8.2.1.11   Transistor List

transistor_list:

```
    "TRANSISTOR"
    { trans_element }*   ;


trans_element :
    element_name
    {          {    part_id }
         |     {    "NPN" |       "PNP" }                    ! Must be present
         |     {    "UBE"         uns_real ["V"] }           ! Default 0.6
         |     {    "TOL+"        uns_real ["%"] }           ! Default 30
         |     {    "TOL-"        uns_real ["%"] }           ! Default 30
         |     {    "BETA"        uns_integer }              ! Default 10
         |     {    "PIN_E"       node_name }
         |     {    "PIN_B"       node_name }
         |     {    "PIN_C"       node_name }
         |     {    "UCE"         uns_real   ["V"] }         ! Default 5.0
         |     {    "I_C1"        uns_real } ["MA"] }        ! Default 9.0
         |     {    "I_C2"        uns_real } ["MA"] }        ! Default 11.0
         |     {    "I_LIM"       uns_real } ["MA"] }        ! Default 100
         |     {    err_msg }
    }*  ;              ! Each parameter only once
```

8.2.1.12   Jumper List

jumper_list:

```
    "JUMPER"
    { jumper_element }*   ;


jumper_element :
    element_name
    {          {    part_id }
         |     {    "PIN_1"       node_name }
         |     {    "PIN_2"       node_name }
         |     {    "OPEN" |       "CLOSED" }          ! Default CLOSED
         |     {    err_msg }
    }*  ;              ! Each parameter only once
```

8.2.1.13   IC List

ic_list:

```
    "IC"
    { ic_element }*   ;


ic_element :


    element_name
    {           {       part_id }
        |       {       "TYPE"          type_name }                 ! Must be present
        |       {       "PINS"          uns_integer }              ! Must be present
        |       {       "PIN_"          uns_integer node_name }*
        |       {       err_msg }
    }*   ;
```

8.2.1.14   Connector List

connector_list:

```
    "CONNECTOR"
    { connector_element }*   ;


connector_element :
    element_name
    {           {       part_id }
        |       {       "TYPE"          type_name }
        |       {       "PINS"          uns_integer }                 ! Must be present
        |       {       "PIN_"          uns_integer  node_name }*
        |       {       err_msg }
    }*   ;
```

8.2.1.15   Black Box List

black_box_list:

```
    "BLACK_BOX"
    { black_box_element }*   ;


black_box_element :


    element_name
    {           {       part_id }
        |       {       "TYPE"          type_name }
        |       {       "PINS"          uns_integer }              ! Must be present
        |       {       "PIN_"          uns_integer node_name }*
        |       {       err_msg }
    }*   ;
```

8.2.1.16 Track List

track_list:

```
"TRACK"
{ track_element }*   ;

track_element :

element_name
{          {      node_name }*               ! At least 2 nodes
      |    {      "PINS"           uns_integer }
      |    {      err_msg }
}*  ;
```

8.2.1.17 Node List

node_list:

```
"NODE"
{ node_element }*   ;

node_element :

element_name
{    connection
}*   ;

connection :

{          "DEV_NAME"          log_name
      {"PIN_" {                "A"
            |                  "B"
            |                  "C"
            |                  "E"
            |                  "T"
            |                  uns_integer
                {
      }*
} ;
```

8.2.1.18 Basic Constructs

```
element_name:

"NAME" log_name ;

part_id :
```

```
    "PART_ID" part_name ;


err_msg :

    "ERR_MSG" message ;


log_name :

    "'"
    { printable }%                ! max. 15 characters
    "'"  ;


part_name :

    "'"
    { printable }%                ! max. 15 characters
    "'"  ;


type_name :

    "'"
    { printable }%                ! max. 15 characters
    "'"  ;


node_name :

    "'"
    { node_chars }%                ! max. 80 characters
    "'"  ;


message :

    "'"
    { printable }%                ! max. 63 characters
    "'"  ;


uns_real :
    { uns_integer [ "." |
                   "." uns_integer ] }
| { "." uns_integer }
[ "E" [ "+" | "-" ] uns_integer ]  ;


uns_integer :
    { dd } %  ;


dd :
    "0" | ... | "9" ;


printable :
    " " | ... | "&" | "(" | ... | "~" ; !all printable ASCII characters with the exception o
```

```
                                                     !quotation mark "'"

            node_chars :
                "A" | ... | "Z" | "a" | ... | "z" | "0" | ... | "9" |
                "_" | "." | "!" | "#" | "$" | "%" | "&" | "*" | "+" | "-" |
                "/" | "\" | ":" | "?" | "@" | "^" | "|" | "~" |
                "(" | ")" | "{" | "}" | "[" | "]" | "<" | ">" ;
```

8.2.1.19   FET List

fet_list:

```
"FET"
{ fet_element }* ;

fet_element :
        element_name
        { { part_id }
                | { "JFET"   |   "DEP_MOSFET |  ENH_MOSFET" } ! Must be present
                | { "N_CHANNEL"  |  "P_CHANNEL" }            ! Must be present
                | { "VGS_OFF"   real_num ["V"] }            ! Default depends on FET type; see
                                                              semantics description
                | { "VGS_ON"    real_num ["V"] }            ! Default depends on FET type; see
                                                              semantics description
                | { "RDS_ON"    uns_real ["OHM"] }          ! Default depends on FET type; see
                                                              semantics description
                | { "PIN_D"      node_name }
                | { "PIN_G"      node_name }
                | { "PIN_S"      node_name }
            | { err_msg }
        }* ; ! Each parameter only once
```

8.2.2   Semantics

8.2.2.1   General

Components of the same type can be combined into lists in the external circuit description. A list of components is started using the related keyword. The individual list elements are separated by the component identifier (name). For this reason the name of the component must always be given at the start of a component description. All other parameters in a component description can be occur in any order, however they can occur a maximum of once.

The component identifier is marked by the keyword NAME and comprises a maximum of 15 characters that are enclosed in single quotation marks.

In all component descriptions except TRACK and NODE, the PART_ID parameter is available for identifying the item number. The item number can comprise a maximum of 15 characters that are enclosed in single quotation marks.

Circuit information on components is started using the keywords PIN_x (x = see syntax for the related component description). The keyword is followed by a node name. The circuit information in the component description is optional. A node name in circuit information comprises a maximum of 80 characters and is placed in parentheses using "".

A node name is only allowed to contain the following characters:

"A"..."Z"	uppercase letter
"a"..."z"	lowercase letter
"0"..."9"	digit
"_"	underscore
"."	decimal point/period
"!"	exclamation mark
"#"	number sign
"$"	dollar sign
"%"	percent
"&"	ampersand
"*"	asterisk
"+"	plus
"-"	minus
"/"	slash
"\"	backslash
":"	colon
"?"	question mark
"@"	at sign
"^"	caret
"|"	vertical bar
"~"	tilde
"("	opening parentheses
")"	closing parentheses
"{"	opening curly brace
"}"	closing curly brace
"["	opening square bracket
"]"	closing square bracket

"<"	opening angle bracket/less than
">"	closing angle bracket/greater than

Using the keyword ERR_MSG you can define for a component a message text that is output in addition to the standard messages on the occurrence of an error. Such messages could, for example, relate to the position of the component or special repair instructions. The message text is allowed to comprise a maximum of 63 characters that are enclosed in single quotation marks.

8.2.2.2 Resistor List

A resistor list is started using the keyword RESISTOR. The individual elements in the list then follow.

VALUE	The keyword VALUE identifies the value for the resistor. A resistor value is represented by a 'uns_real', followed optionally by unit of measure information K (Kilo) or M (Mega). The numerical representation is not allowed to exceed 6 characters. The unit of measure information OHM is only used for improved readability, it is optional.
TOL+, TOL-	These keywords identify the tolerance of the resistor in percent. The tolerance of the resistor is represented by a 'uns_real'. The numerical representation is not allowed to exceed 3 characters, the unit of measure information % is only used for improved readability, it is optional.
I_LIM	This parameter specifies the maximum current in milliamps that is allowed to be applied to the resistor during the test. The current value is represented by a 'uns_real'. The numerical representation is not allowed to exceed 5 characters. The unit of measure information MA (milliamp) is only used for improved readability, it is optional. If the parameter is not listed, the default value of 100 MA is assumed.

Example:

BDL RESISTOR

```
RESISTOR
NAME 'R1'  PART_ID 'Part_no_r1'
           VALUE 10.0 K OHM TOL+ 5% TOL- 5%
           PIN_1 'Node_r1.1'
           PIN_2 'Node_r1.2'
           I_LIM 10  MA
           ERR_MSG 'Fault at location A-17'
NAME 'R2'  PART_ID 'Part_no_r2'
           VALUE 1.4
           PIN_1 'Node_r2.1'
           PIN_2 'Node_r2.2'
```

8.2.2.3   Variable Resistor List

A list of variable resistors is started with the keyword VAR_RES. The individual elements in the list then follow.

VALUE	The parameter VALUE defines the maximum resistance of the component. The information is given in a similar manner to that for RESISTOR (see section 8.2.1.1).
TOL+, TOL-	Tolerance of the variable resistor. The same information as for RESISTOR apply (see section 8.2.1.1).
I_LIM	This parameter specifies the maximum current in milliamps that is allowed to be applied to the resistor during the test. The same information as for RESISTOR applies (see section 8.2.1.1).
SETUP	This parameter defines the presetting for the resistor. The resistor value is represented by a 'uns_real', followed optionally by unit of measure information K (Kilo) or M (Mega). The numerical representation is not allowed to exceed 6 characters. The unit of measure information OHM is only used for improved readability, it is optional.
ADJ/FIX	The parameter **ADJ/FIX** is not required in and is only present for reasons of compatibility with TSS. If the parameter is given in , it is only allowed to be **FIX**.

Example:

BDL VAR_RES

```
VAR_RES
NAME 'VR1'  PART_ID 'Part_vr1'
            VALUE 5.0 K OHM TOL+ 5% TOL- 5%
            SETUP 1.45 KOHM
            PIN_1 'Node_vr1.1'
            PIN_2 'Node_vr1.2'
            I_LIM 50 MA
            ERR_MSG 'Fault at location C-5'
NAME 'VR2'  PART_ID 'Part_vr2'
            VALUE 1.0 M
            SETUP 400 K
            I_LIM 50 MA
            ERR_MSG 'Fault at location R-43'
```

8.2.2.4   Potentiometer List

A list of potentiometers is started using the keyword POTI. The individual elements in the list then follow.

Parameters

The same information as for VARIABLE RESISTOR applies (see section Chapter 8.2.2.2, "Resistor List", on page 162).

R_2_3	The keyword SETUP is replaced by the keyword R_2_3 for improved understanding. The parameter R_2_3 defines the presetting for the resistor between the pins PIN_2 (wiper) and PIN_3.

Example:

BDL POTI

```
POTI
NAME 'Poti1' PART_ID 'Part_p1'
             VALUE 5.0 K OHM TOL+ 5% TOL-5%
             R_2_3 1.45 K OHM
             PIN_1 'Node_p1.1'
             PIN_2 'Node_p1.2' 'Node_p1.3'
             I_LIM 50 MA
             ERR_MSG 'Fault at location
             G-29'
NAME 'Poti2' PART_ID 'Part_p2'
             VALUE 1.0 M
             R_2_3 400 K
             PIN_1 'Node_p1.1'
             PIN_2 'Node_p1.2' 'Node_p1.3'
             I_LIM 50 MA
             ERR_MSG 'Fault at location I-2'
NAME 'Poti3' R_2_3 400 K
             VALUE 2 K
             PIN_1 'Node_p1.1'
             PIN_2 'Node_p1.2' 'Node_p1.3'
NAME 'Poti4' VALUE 5.0 K TOL+ 10% TOL- 10%
             R_2_3 1 K
             PIN_1 'Node_p1.1'
             PIN_2 'Node_p1.2' 'Node_p1.3'
```

8.2.2.5 Resistor Array List

A list of resistor networks is started using the keyword RESISTOR_ARRAY. The individual elements in the list then follow.

ARRAY	The keyword ARRAY identifies the size of the resistor network, i.e. the number of resistors in the network. The size of the array is limited to 127 resistors. The number of resistors implicitly defines the number of pins on the network. Here the following applies: ARRAY n corresponds to n resistors and n +1 pins. PIN_1 is the common pin for the network.
X	The number of resistors is followed by the keyword 'X', followed by the resistor value in Ω, kΩ or MΩ. The information on the resistor value is given as 'uns_real', followed optionally by the unit of measure information K or M. The entry of the code OHM is optional, it is only used for improved readability.
TOL+, TOL-	These keywords identify the tolerance of the resistor in percent. The tolerance of the resistor is represented by a 'uns_real'. The numerical representation is not allowed to exceed 3 characters, the unit of measure information % is only used for improved readability, it is optional.
I_LIM	This parameter specifies the maximum current in milliamps that is allowed to the applied to the resistor during the test. The value for the current is represented by a 'uns_real'. The numerical representation is not allowed to exceed 5 characters. The unit of measure information MA (milliamp) is only used for improved readability, it is optional. If the parameter is not listed, the default value of 100 MA is assumed.

Example:

BDL RESISTOR_ARRAY

```
RESISTOR_ARRAY
    NAME 'R137'    PART_ID '111,222'
    ARRAY 11 X 1 OHM
    TOL+ 10   TOL- 10
    I_LIM 100.0 MA
    PIN_1   'A1_1'
    PIN_2   'A1_2'
    PIN_3   'A1_3'
    PIN_4   'A1_4'
    PIN_5   'A1_5'
    PIN_6   'A1_6'
    PIN_7   'A1_7'
    PIN_8   'A1_8'
    PIN_9   'A1_9'
    PIN_10  'A1_10'
    PIN_11  'A1_11'
    PIN_12  'A1_12'
    ERR_MSG 'ERROR in R137'
```

8.2.2.6   Capacitor List

A list of capacitors is started using the keyword CAPACITOR. The individual elements in the list then follow.

VALUE	The parameter VALUE defines the value of the capacitor. The value is represented by a 'uns_real' followed optionally by unit of measure information M (Milli), U (Micro), N (Nano), P (Pico). The numerical representation is not allowed to exceed 6 characters. The unit of measure information F is only used for improved readability, it is optional.
TOL+, TOL-	These keywords identify the tolerance of the component in percent. The tolerance is represented by a 'uns_real'. The numerical representation is not allowed to exceed 3 characters. The unit of measure information % is only used for improved readability, it is optional. If a parameter does not occur in the component description, then a default tolerance of 10 % is assumed.
FREQ	This parameter specifies the test frequency in KHZ at which the test on the capacitor is to be performed. The information on a test frequency is optional. The test frequency is represented by a 'uns_real'. The numerical representation is not allowed to exceed 4 characters. Only the frequencies 0.1 KHZ, 1.0 KHZ and 10.0 KHZ are allowed. To avoid errors in the real number arithmetic, variations of up to 10 % from the setpoint in each case is allowed. The unit of measure information KHZ is only used for improved readability, it is optional.

Example:

BDL CAPACITOR

```
CAPACITOR
NAME 'C1'  PART_ID 'Part_vr1'
           VALUE 5.0 N F TOL+ 15% TOL- 15%
           PIN_1 'Node_c1.1'
           PIN_2 'Node_c1.2'
           FREQ 1.0 KHZ
           ERR_MSG 'Fault at location A-32'
NAME 'C2'  VALUE 0.5 M TOL+ 5% TOL- 5%
           PIN_1 'Node_c2.1'
           PIN_2 'Node_c2.2'
           PART_ID 'Part_c2'
```

8.2.2.7   Pol. Capacitor List

A list of polarized capacitors with poles is started using the keyword POL_CAP. The individual elements in the list then follow.

The same information as for CAPACITOR applies (see section).

Example:

BDL POL_CAP

```
POL_CAP
NAME 'ELKO1'  PART_ID 'Elko_1'
              VALUE 5.0 N F TOL+ 15% TOL- 15%
              PIN_A 'Node_pc1.1' PIN_C 'Node_pc1.2'
              FREQ 1.0 KHZ
              ERR_MSG 'Fault at location F-22'
NAME 'ELKO2'  VALUE 0.5 M TOL+ 5% TOL- 5%
              PIN_A 'Node_pc2.1' PIN_C 'Node_pc2.2'
              PART_ID Elko.-2'
```

PIN_A	= Anode	= positive pole of the polarized capacitor
PIN_C	= Cathode	= negative pole of the polarized capacitor

8.2.2.8 Inductor List

A list of inductors is started using the keyword INDUCTOR. The individual elements in the list then follow.

VALUE	The parameter VALUE defines the inductance of the component. The value is represented by a 'uns_real', followed optionally by unit of measure information M (Milli). The numerical representation is not allowed to exceed 6 characters. The unit of measure information H is only used for improved readability, it is optional.
TOL+, TOL-	These keywords identify the tolerance of the component in percent. The tolerance is represented by a 'uns_real'. The numerical representation is not allowed to exceed 3 characters. The unit of measure information % is only used for improved readability, it is optional. If a parameter does not occur in the component description, then the default tolerance of 20 % is assumed.
DC_RES	This parameter defines the DC resistance of the component in OHM. The resistance value is represented by a 'uns_real'. The numerical representation is not allowed to exceed 5 characters. The unit of measure information OHM is only used for improved readability, it is optional. If a parameter does not occur in the component description, then the default resistance of 1.0 OHM is assumed.

R_TOL+, R_TOL-	These keywords identify the tolerance on the DC resistance of the component in percent. The tolerance is represented by a 'uns_real'. The numerical representation is not allowed to exceed 3 characters. The unit of measure information % is only used for improved readability, it is optional. If a parameter does not occur in the component description, then the default tolerance of 30 % is assumed.
FREQ	This parameter specifies the test frequency KHZ at which the test is to be performed on the component. The information on a test frequency is optional. The test frequency is represented by a 'uns_real'. The numerical representation is not allowed to exceed 4 characters. Only the frequencies 0.1 KHZ, 1.0 KHZ and 10.0 KHZ are allowed.

Example:

BDL INDUCTOR

```
INDUCTOR
NAME 'I1'  PART_ID 'Part_i1'
           VALUE 50.0 M H TOL+ 10% TOL- 10%
           DC_RES 20 OHM R_TOL+ 15% R_TOL- 15%
           FREQ 10.0 KHZ
           PIN_1 'Node_i1.1' PIN_2 'Node_i1.2'
           ERR_MSG 'Fault at location K-10'
NAME 'I2'  PART_ID 'Part-i2'
           VALUE 0.1 TOL+ 5% TOL- 5%
           PIN_1 'Node_i2.1' PIN_2 'Node_i2.2'
NAME 'I3'  VALUE 0.02 H TOL+ 10 TOL- 10
           DC_RES 5 R_TOL+ 10% R_TOL- 10%
           PIN_1 'Node_i3.1' PIN_2 'Node_i3.2'
           ERR_MSG 'Fault at location D-23'
```

8.2.2.9 Diode List

A list of diodes is started using the keyword DIODE. The individual elements in the list then follow.

VALUE	The parameter VALUE defines the knee voltage of the diode in V. If the parameter is missing in the component description, then a knee voltage of 0.7 V is assumed. The value is represented by a 'uns_real'. The numerical representation is not allowed to exceed 3 characters. The unit of measure information V is only used for improved readability, it is optional.
TOL+, TOL-	These keywords identify the tolerance on the knee voltage in percent. The tolerance is represented by a 'uns_real'. The numerical representation is not allowed to exceed 3 characters. The unit of measure information % is only used for improved readability, it is optional. If parameter does not occur in the component description, then the default tolerance of 25 % is assumed.
I_DCS	This parameter defines the maximum permissible total current during the test on the diode in milli-amps. The value is represented by a 'uns_real'. The numerical representation is not allowed to exceed 5 characters. The unit of measure information MA is only used for improved readability, it is optional. If the parameter is missing in the component descrip-tion, then a current limit of 100 MA is assumed.
I_DIO	This parameter defines the maximum permissible diode current in milliamps. The value is represented by a 'uns_real'. The numerical information is not allowed to exceed 4 characters. The unit of measure information MA is only used for improved readability, it is optional. If the parameter is missing in the com-ponent description, then a maximum diode current of 10 MA is assumed.

Example:

BDL DIODE

```
DIODE
NAME 'D1'  PART_ID 'Part_d1'
           VALUE 0.4 V TOL+ 15% TOL- 15%
           I_DCS 50 MA
           I_DIO 5.0 MA
           PIN_A 'Node_d1.1'
           PIN_C 'Node_d1.2'
           ERR_MSG 'Fault at location H-19'
NAME 'D2'  PART_ID 'Part_d2'
           VALUE 0.4
           I_DCS 20 MA
           PIN_A 'Node_d2.1'
           PIN_C 'Node_d2.2'
```

PIN_A	= Anode
PIN_C	= Cathode

8.2.2.10 LED List

A list of light emitting diodes is started using the keyword LED. The same information as for diodes applies (see section Chapter 8.2.2.9, "Diode List", on page 168), only the default value for VALUE here is 1.2 V.

8.2.2.11 Zener Diode List

A zener diode test is only generated if a R&S TS-PSU or R&S TS-PSU12 module is configured in the PHYSICAL.INI file. Otherwise, only a diode forward test is inserted.

A list of zener diodes is started using the keyword Z_DIODE. The individual elements in the list then follow.

VALUE	The parameter VALUE defines the nominal voltage of the zener diode in V. The value is represented by a 'uns_real'. The numerical information is not allowed to exceed 4 characters. The unit of measure information V is only used for improved readability, it is optional.
TOL+, TOL-	These keywords identify the tolerance on the nominal voltage in percent. The tolerance is represented by a 'uns_real'. The numerical representation is not allowed to exceed 3 characters. The unit of measure information % is only used for improved readability, it is optional. If a parameter does not occur in the component description, then a default tolerance of 10 % is assumed.
I_Z	This parameter defines the zener current of the diode in milliamps. The information on the zener current is optional. The value is represented by a 'uns_real'. The numerical information is not allowed to exceed 4 characters. The unit of measure information MA is only used for improved readability, it is optional.
R_Z	This parameter defines the zener resistance of the diode in OHM. The information on the zener resistance is optional. The value is represented by a 'uns_real'. The numerical information is not allowed to exceed 4 characters. The unit of measure information OHM is only used for improved readability, it is optional.

Example:

BDL Z_DIODE

```
Z_DIODE
NAME 'Z1'  PART_ID 'Part_z1'
           VALUE 10.0 V TOL+ 15% TOL- 15%
           I_Z 5 MA
           R_Z 1.0 OHM
           PIN_A 'Node_z1.1'
           PIN_C 'Node_z1.2'
           ERR_MSG 'Fault at location E-9'
NAME 'Z2'  PART_ID 'Part_z2'
           VALUE 5.0 V
           I_Z 10 MA
           PIN_A 'Node_z2.1'
           PIN_C 'Node_z2.2'
```

PIN_A	= Anode
PIN_C	= Cathode

8.2.2.12 Transistor List

A list of transistors is started using the keyword TRANSISTOR. The individual elements in the list then follow.

A transistor beta test is only generated if a R&S TS-PSU or R&S TS-PSU12 module is configured in the PHYSICAL.INI file. Otherwise, only a simple transistor test is inserted

NPN, PNP	These parameters identify the type of the transistor. The two parameters are mutually exclusive.
UBE	The parameter UBE defines the base-emitter voltage of the transistor in the forward bias region in V. If the parameter is missing, then a base-emitter voltage of 0.6 V is assumed. The value is represented by a 'uns_real'. The numerical information is not allowed to exceed 4 characters. The unit of measure information V is only used for improved readability, it is optional.
TOL+, TOL-	These keywords identify the tolerance on the base-emitter voltage in percent. The tolerance is represented by a 'uns_real'. The numerical representation is not allowed to exceed 3 characters. The unit of measure information % is only used for improved readability, it is optional. If a parameter does not occur in the component description, then the default tolerance of 30 % is assumed.

BETA	This parameter defines the minimum current gain of the transistor. If the parameter is missing, a gain of 10 is used. The gain is represented by a 'uns_integer'. The numerical information is not allowed to exceed 3 characters.
UCE	This parameter defines the collector-to-emitter voltage in V for the transistor test. If the parameter is missing, a voltage of 5.0 V is used. The value is represented by a 'uns_real'. The numerical information is not allowed to exceed 4 characters. The unit of measure V is only used for improved readability, it is optional.
I_C1, I_C2	These two parameters define the collector-to-emitter currents in milliamps at which the transistor gain is to be measured in the test. The default settings for the two currents are 9.0 MA for I_C1 and 11.0 MA for I_C2. The value is represented by a 'uns_real'. The numerical information is not allowed to exceed 5 characters. The unit of measure MA is only used for improved readability, it is optional.
I_LIM	This parameter specifies the maximum current from the DC voltage source (DCS) in milliamps that is allowed to be applied during the test on the transistor. The value for the current is represented by a 'uns_real'. The numerical representation is not allowed to exceed 5 characters. The unit of measure information MA (milliamp) is only used for improved readability, it is optional. If the parameter is not listed, the default value of 100 MA is assumed.

Example:

BDL TRANSISTOR

```
TRANSISTOR
NAME 'T1' PART_ID 'Part_t1'
        PNP
        UBE 0.8 V TOL+ 20% TOL- 20%
        I_LIM 50 MA
        PIN_E 'Node_t1.e' PIN_B 'Node_t1.b' PIN_C 'Node_t1.c'
        ERR_MSG 'Fault at location D-4'
 NAME 'T2' PART_ID 'Part-t2'
        NPN
        UBE 0.4 V
        PIN_E 'Node_t2.e' PIN_B 'Node_t2.b' PIN_C 'Node_t2.c'
```

8.2.2.13 Jumper List

A list of jumpers is started using the keyword JUMPER. The individual elements in the list then follow.

OPEN, CLOSED	The parameters define whether the jumper is OPEN or CLOSED. The two parameters are mutually exclusive. If the parameter is missing, the CLOSED state is assumed.

Example:

BDL

```
JUMPER
NAME 'JMP1'  PART_ID 'Part_jmp1'
             PIN_1 'Node_J1.1' PIN_2 'Node_J1.2' OPEN
             ERR_MSG 'Fault at location C-15'
NAME 'JMP2'  PART_ID 'Part_jmp2'
             PIN_1 'Node_J2.1' PIN_2 'Node_J2.2'
```

8.2.2.14   IC List

A list of ICs is started using the keyword IC. The individual elements in the list then follow.

TYPE	This parameter defines the type of IC. A type identifier comprises a maximum of 15 characters and is placed in parenthesis using "". Within the type identifier there must not be any "" or spaces.
PINS	This parameter defines the number of pins on the IC. The number is represented by a 'uns_integer'. The numerical representation must not exceed 3 characters. The number of pins is not allowed to exceed 999. In the circuit information, the related pin number is not allowed to exceed the number of pins on the IC.

Example:

BDL IC

```
IC
NAME 'IC1' PART_ID 'Part-IC1'
        TYPE '74LS02'
        PINS 14
        PIN_1 'IC1.1' PIN_2 'IC1.2' PIN_3 'IC1.3' PIN_4 'nc'
        PIN_5 'nc' PIN_6 'IC1.6'  PIN_7 'VCC' PIN_8 'IC1.8'
        PIN_9 'nc' PIN_10 'nc' PIN_11 'IC1.11'
        PIN_12'nc' PIN_13 'IC1.13' PIN_14 'GND'
```

ICs and the connections on the ICs are taken into account during the **Contact** and **Short** tests. IC pins are specially protected by the Automatic Test Generator ATG.

8.2.2.15   Connector List

A list of connectors is started using the keyword CONNECTOR. The individual elements in the list then follow.

TYPE	This parameter defines the type of connector. The information on the type is optional here. A type identifier comprises a maximum of 15 characters and is placed in parenthesis using "". Within the type identifier there must not be any "" or spaces.
PINS	This parameter defines the number of pins on the connector. The number is represented by a 'uns_integer'. The numerical representation must not exceed 3 characters. In the circuit information, the related pin number must not exceed the number of pins on the connector.

Example:

BDL CONNECTOR

```
CONNECTOR
NAME 'CON-1'  PART_ID 'Part-CON-1'
            PINS 4
            PIN_1 'CN1.1' PIN_2 'CN1.2'
            PIN_3 'CN1.3' PIN_4 'CN1.4'
```

8.2.2.16 Black Box List

A list of black boxes is started using the keyword BLACK_BOX. The individual elements in the list then follow. A black box can be used to insert a placeholder test for user-defined test methods.

TYPE	This parameter defines the type of black box. The information on the type is optional here. A type identifier comprises a maximum of 15 characters and is placed in parenthesis using "". Within the type identifier there must not be any "" or spaces.
PINS	This parameter defines the number of pins on the black box. The number is represented by a 'uns_integer'. The numerical representation must not exceed 3 characters. The number of pins is not allowed to exceed 999. In the circuit information, the related pin number is not allowed to exceed the number of pins on the black box.

Example:

BDL BLACK_BOX

```
BLACK_BOX
NAME 'Bl1' PART_ID 'Part-bl-1'
        TYPE 'BOX-1'
        PINS 14
        PIN_1 'BL1.1' PIN_2 'BL1.2' PIN_3 'BL1.3'
        PIN_4 'nc' PIN_5 'nc' PIN_6 'BL1.6'
        PIN_7 'VCC' PIN_8 'BL1.8' PIN_9 'nc'
        PIN_10 'nc' PIN_11 'BL1.11' PIN_12'nc'
        PIN_13 'BL1.13' PIN_14 'GND'
        ERR_MSG 'Fault at location L-16'
```

Black boxes and the connections on black boxes are taken into account during the **Contact** and **Short** tests.

In addition, a contact test is inserted for the black box as a placeholder.

8.2.2.17 Track List

A list of tracks is started using the keyword TRACK. The individual elements in the list then follow.

The parameter is a list of node names that are connected to the track. At least 2 nodes must be allocated to the track.

Example:

BDL example TRACK

```
TRACK
NAME 'TRACK-1'
     PINS 4
     'Node_1.1' 'Node_1.2' 'Node_1.3' 'Node_1.4'
     ERR_MSG 'Fault at location B-3 to B-22'
NAME 'TRACK-2'
     PINS 2
     'Node_2.1' 'Node_2.2'
```

8.2.2.18 Node List

A list of nodes is started using the keyword NODE. The individual elements in the list then follow.

DEV_NAME	This parameter identifies the logical name of the component that is connected to this node.

Example:

BDL example NODE

```
NODE

NAME 'NODE1'
     DEV_NAME 'IC' PIN_1
     DEV_NAME 'IC' PIN_2
     DEV_NAME 'IC' PIN_3
     DEV_NAME 'Transistor-1' PIN_A
     DEV_NAME 'LED_1' PIN_A

NAME 'NODE2'
     DEV_NAME 'LED_1' PIN_C
     DEV_NAME 'Transistor-1' PIN_B

NAME 'NODE3'
     DEV_NAME 'Transistor-1' PIN_C
```

8.2.2.19 FET List

A list of transistors is started using the keyword FET. The individual elements in the list then follow.

JFET, **DEP_MOSFET,** **ENH_MOSFET**	Defines the type of field-effect transistor (FET). Junction field-effect transistors (junction-fet; JFET) and metal oxide semiconductor field-effect transistors (MOSFET) are distinguished. For MOSFETs there are additionally the depletion-mode MOSFET (DEP_MOSFET) and enhancement-mode MOSFET (ENH_MOSFET) types.
N_CHANNEL, **P_CHANNEL**	Defines the channel type of a field-effect transistor.

VGS_OFF	Defines the gate-source voltage at which the transistor insulates (FET). If the parameter is omitted, the following default values will be used:
	
	JFET
	  N_CHANNEL: -5 V
	  P_CHANNEL: 5 V
	DEP_MOSFET
	  N_CHANNEL: -5 V
	  P_CHANNEL: 5 V
	ENH_MOSFET
	  N_CHANNEL: 0 V
	  P_CHANNEL: 0 V

VGS_ON	Defines the gate-source voltage at which the transistor conducts. If the parameter is omitted, the following default values will be used:
	JFET
	N_CHANNEL: 0 V
	P_CHANNEL: 0 V
	DEP_MOSFET
	N_CHANNEL: 0 V
	P_CHANNEL: 0 V
	ENH_MOSFET
	N_CHANNEL: 5 V
	P_CHANNEL: -5 V
RDS_ON	Defines the drain-source resistance on condition switched on. If the parameter is omitted, the following default values will be used:
	JFET: 1000 Ohm
	DEP_MOSFET, ENH_MOSFET: 0.1 Ohm

8.3 Important Additional Information

8.3.1 Node List

The Automatic Test Generator ATG generates the relevant node list from the component data if no node list is given in the BDL file.

The node list must be completely and stable if it is given in the BDL file. The Automatic Test Generator ATG compares only the content of the BDL file data with the node list entries. When there differences will be detected the ATG generates an error message.

8.3.2 Treatment of Specific Pins

When naming nodes, it must be observed that the following 2 node names will be treated separately by Automatic Test Generator ATG because they are very important for a correct test generation.

VCC...	If the node name initials are VCC or vcc it means that it is a power supply node.
GND...	If the node name initials are GND or gnd it means that it is a ground node.

VCC and GND nodes will particularly be considered on capacitor discharging (Test Step Discharge by ATG).

9 Automatic Test Generation with ATG

9.1 Function

Using the Automatic Test Generator ATG utility, an executable ICT program that can be run in Enhanced Generic Test Software Library R&S EGTSL is generated from circuit description and the description of the hardware available. Figure 9-1 shows the general sequence for automatic test generation using the Automatic Test Generator ATG.
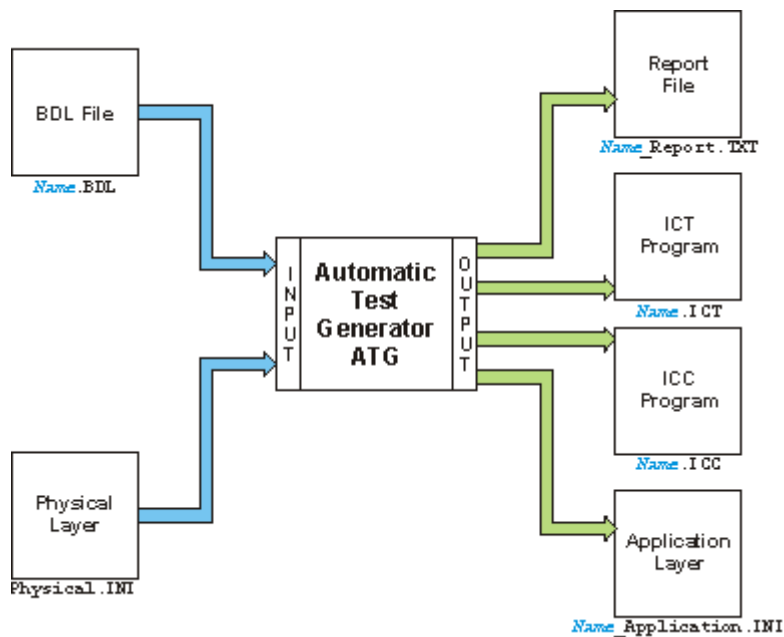
Figure 9-1: ATG sequence

The circuit description must be available in the BDL circuit description language (on this topic, see also Chapter 8.2, "External File Format (BDL)", on page 151). The test hardware available is described using the Physical Layer Configuration File (on this topic, see also Chapter 7.2, "PHYSICAL.INI", on page 140). When the ATG is started, the file names for the BDL file (`Name.BDL`) and the file name for the Physical Layer Configuration File (`Physical.INI`) must be given.

Following the generation process, the ATG creates the following files:

- `Name_Report.TXT`
  Report with the data on the generation process
- `Name.ICT`
  ICT program for execution in R&S EGTSL
- `Name.ICC`
  ICC program for execution in R&S IC-Check
- `Name_Application.INI`

Application Layer Configuration File for the execution of the ICT program generated in R&S EGTSL

Any errors that have occurred during the generation process are indicated in the report using error messages and warning messages. The ICT program and the ICC program is only created when there are no more error messages.

The result of the automatic test generation is essentially defined by realistic entry of the circuit description. The circuit description must correspond to the actual topology of the circuit. Otherwise, it cannot be ensured that the automatically generated ICT and ICC programs will work.

ⓘ The circuit description (BDL object) from which the automatically generated ICT and ICC programs are to be generated must be free of logical data entry errors and the circuit entered must correspond as far as possible to reality.

9.2 Starting the Automatic Test Generator ATG

Automatic Test Generator

Start the Automatic Test Generator ATG using **"Start -> Programs -> GTSL -> Automatic Test Generator"**.

The following buttons have the same function in all windows of the ATG:
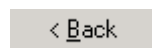
< Back	The entries made in the current window are discarded and the previous ATG window is opened.
Next >	The entries made are applied and the next ATG window is opened.
Cancel	The ATG is stopped. All entries made are discarded.
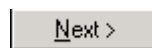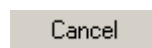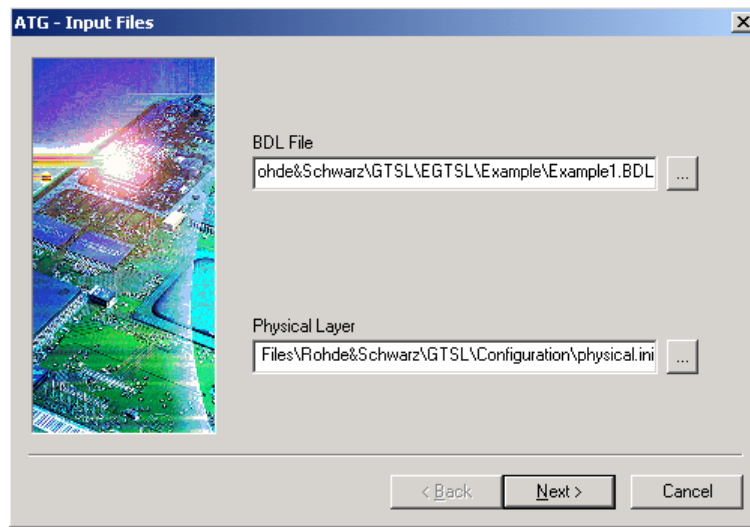
Figure 9-2: ATG - input files

"BDL File"	The path and the file name for the BDL input file required by the ATG are displayed in this field. The information can be edited in the field.
"Physical Layer"	The path and file name of the Physical Layer Configuration File required by the ATG are displayed in this field. The information can be edited in the field.
...	Opens the standard Windows dialog box for opening files. The save path and the file name for the input files can be selected in this dialog box. The file selected including the save path are displayed in the related field.
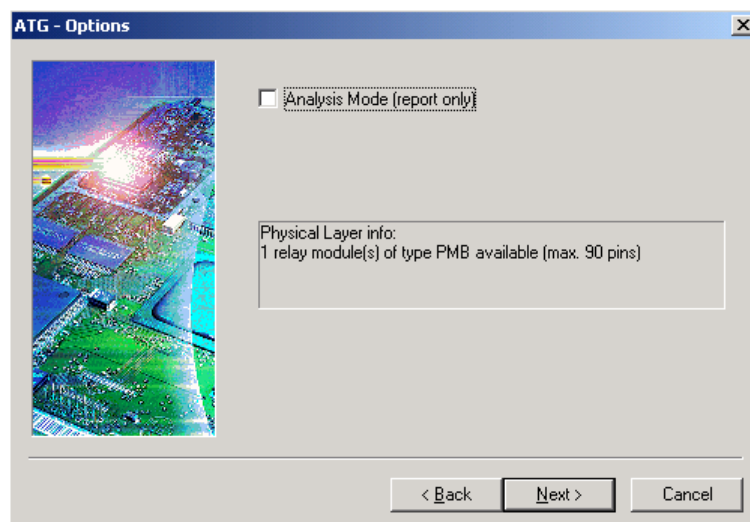
Figure 9-3: ATG - Options

"Analysis Mode (report only)"

With this function activated the entire generation process is performed. However only a report file with the data on the generation process is created. No ICT program, no ICC program and no Application Layer Configuration File are created.

"Physical Layer Info:"

The hardware information given in the Physical Layer Configuration File is displayed in this field.
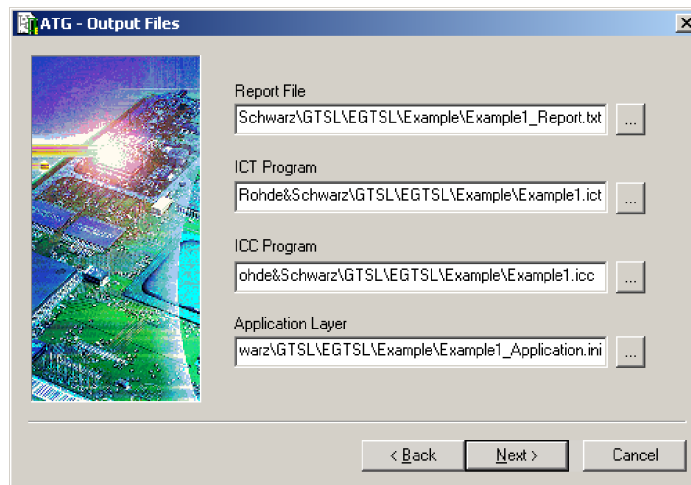
Figure 9-4: ATG - Output Files

Report File

The save path and the file name of the report file generated by the ATG are displayed in this field. The information can be edited in the field.

ICT Program

The save path and the file name of the ICT program generated by the ATG are displayed in this field. The information can be edited in the field.

ICC Program

The save path and the file name of the ICC program generated by the ATG are displayed in this field. The information can be edited in the field.

Application Layer

The save path and the file name of the Application Layer Configuration File are displayed in this field. The information can be edited in the field.

Opens the standard Windows dialog box for opening or adding files. The save path and the file name for the output files can be selected using this dialog box. It is also possible to add a new output file. The file selected or added is displayed in the related field.
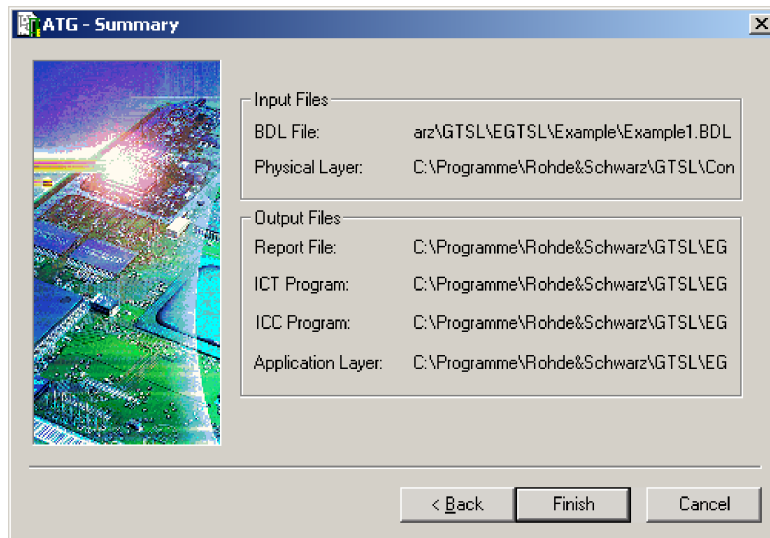
Figure 9-5: ATG - Summary

All information on the input files and the output files is displayed once again in this window for a last check.

The ATG generation process is started.
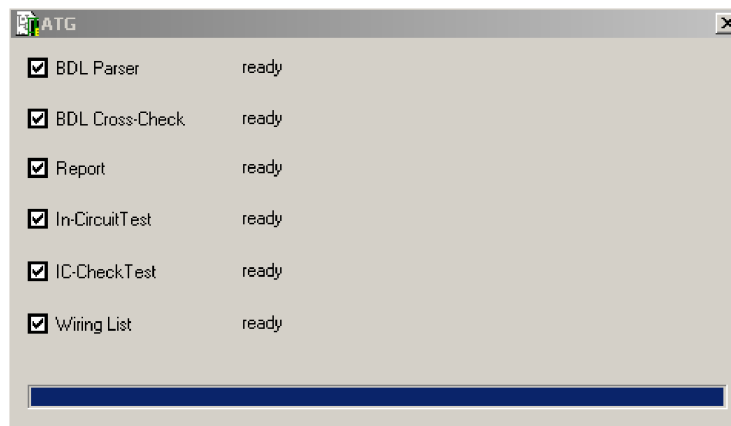
Figure 9-6: ATG Process

The generation process progress is displayed with the individual steps. If an error should occur in a step during the generation process, execution is stopped immediately. A report file with the related error message is generated when the generation is stopped.

If no errors occur during the generation process, the three output files are generated by the ATG. The report file generated can be displayed directly in both cases (error, no error).
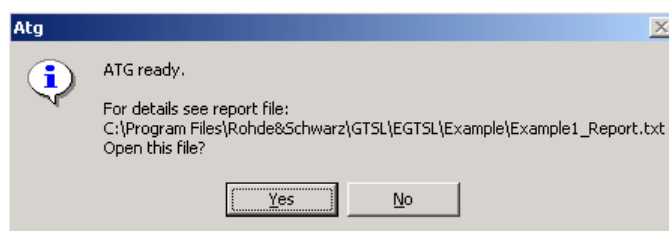
Figure 9-7: ATG Finish

9.3 Output Files

9.3.1 ICT Program

9.3.1.1 General

The objective of the Automatic Test Generator ATG is to generate an ICT program for the in-circuit test of a circuit board.

The ICT program to be generated by the ATG contains test steps for discharging capacitors, the contact test, the short-circuit and connection test and the analog test in a specific order.

The automatically generated test program can be run immediately after generation. A prerequisite is an error-free circuit description (BDL object).

A bed of nails adapter is required for the test.

9.3.1.2 Layout of the Test Program

The layout of the ICT program generated automatically is defined by a structure anchored in the ATG. In principle, the fully generated test program has the following layout:

1. Capacitor discharging (Discharge)

2. Contact test (Contact)

3. Continuity and short-circuit test (Short)

4. Group of analog tests (In-Circuit)

9.3.1.3 State of the Automatically Generated Test Program

The vast majority of the automatically generated test steps can be used without further changes. There are, however, certain topological conflicts that cannot be solved by the ATG that must be edited when debugging the ICT program.

These test conflicts are prepared by the ATG such that they can be addressed by the user as easily as possible. The aids generated by the ATG for this purpose comprise the comments in the ATG report and test proposals in the ICT program that can be applied by the user after review.

Test conflicts can, for example, be:

- Direct parallel component paths (SHUNTS)
- Very small nominal values
- Very small parallel resistors
- Forward biased, parallel diode paths
- Hazards for components

All the above mentioned problems are detected by the network analysis in the ATG and processed accordingly.

9.3.1.4 Program Groups Generated

Capacitor discharging (Discharge)

The capacitor discharging is intended to prevent discharge currents from capacitors producing incorrect measurements or damaging the relays in the switching matrix.

The pins in the **"Reference Pin List"** are short-circuited at the start of the discharging. The pins in the **"Test Pin List"** are individually discharged to the reference pin potential and then connected to the potential of the **"Reference Pin List"**.

The ATG uses the following strategy for the layout of the pin lists:

- The Reference Pin List contains the ground nodes with the largest number of component connections (ground nodes always start with "GND" or "gnd").
- The Test Pin List contains
    - All other ground nodes.
    - All nodes that start with "VCC" or "vcc".
    - All nodes between which there is a total capacitance of 8 µF or more.

Contact Test (Contact)

The contact test is intended to exclude the possibility of missing nail contacts resulting in inappropriate error messages.

In the contact test generated by the ATG, all nodes on the circuit board for which there is a corresponding entry in the related BDL object, i.e. in the table of nail allocations, are always checked. During this process the connection to all other nodes in their entirety is tested. The resistance between a node and all others must not exceed the value of 1 MΩ (default value). This is defined by the parameter **"Resistor Limit"** in the contact test.

If there are isolated nodes in the circuit to be tested, i.e. nodes that do not have a connection to any other point on the circuit board, these must be removed from the contact test using the **All but not** function.

Continuity and Short-circuit Test (Short)

For closed jumpers and tracks connection tests are integrated into the ICT program by the ATG. During this process the jumper and track nodes are tested for the existence of the nominal connection; the limit for a connection that is still valid is preset in the **"Resistor Limit"** parameter to 10 Ω.

For open jumpers and connectors short-circuit tests are integrated into the ICT test by the ATG. In contrary to the connection test, in this case the nodes given are tested for mutual short-circuits. If a short-circuit is found, i.e. a connection with less than 10 Ω (default value), the result will be a FAIL.

A global short-circuit test is always generated over all nodes on the unit under test by the ATG in the test program. During the generation of the global short-circuit test, nodes that have a nominal connection (e.g. closed jumpers, tracks or coils) are recorded by the ATG and removed from the short-circuit test using the **All but not** function.

The Automatically Generated Analog Test

Automatic test generation is possible for the following components that can be given in the circuit description language BDL.

Table 9-1: Possible analog tests

Description in BDL	Test method
Resistor	Resistor or Continuity
Resistor Array	
Potentiometer	Resistor
Variable Resistor	
Capacitor	Impedance or Short
Pol. Capacitor	Impedance or Short
Inductor	Impedance or Continuity
Diode	Diode
LED	
Z_Diode	ZenerDiode or Diode
Transistor	TransistorBeta or Transistor
Jumper	Continuity or Short
IC	No test
Connector	Short
Black Box	Contact (placeholder)
Track	Continuity
FET	FET (field-effect transistor)

Tests on the same analog component types are combined by the ATG and generated together. Thus, in an automatically generated test you will find all resistor tests, capaci-

tor tests, etc. combined into groups. The order of the individual tests within a group corresponds to the entry in BDL.

9.3.1.5 Automatic Determination of the Guard Points in the Analog Test

Measuring errors can occur due to currents flowing parallel to the component paths to be tested; these errors can degrade the effectiveness of an analog test. To prevent this problem occurring, the **Guarding** process is used in the analog in-circuit test. During this process specific nodes (guard points) on the circuit are selected and brought to a potential with the measuring point. In this way it is ensured that the current measured corresponds to that which is flowing through the component path to be measured.

The relevant guard points are determined automatically by the ATG. During this process, those nodes that are in close proximity to the component to be tested are preferably accepted as guard points. This condition is met,

- When the topological distance between measuring and stimulation point is not larger than one branch and
- When the guard point is within a feedback branch to the component to be measured, that is within the same cluster.

9.3.1.6 Safety Against Destruction of Components During the Test

During the circuit analysis and the editing of analog component tests, the ATG monitors the following possible hazards:

- Application of a voltage > 0.7 V to an IC that is not supplied with power.
- Possible forward biasing of a diode in the vicinity of an IC during the test on a component connected to the diode.
- Reverse polarization of an electrolytic capacitor during a test on an adjacent component by more than 0.5 V.

If a hazard is found, the corresponding test configuration is skipped by the ATG and, if possible, a different configuration selected. A further possibility is the listing of such hazards in the ATG report with the corresponding comments.

9.3.1.7 Taking Account of Topological Problems

The ATG detects and processes the following circuit situations:

- Direct and indirect parallel circuit paths
- Orientation of parallel diode paths

During the generation of an analog component test, the measuring configurations that permit measurement on the component despite a topological problem are initially preferred. In the case of a parallel diode, e.g., the direction in which the voltage is applied is changed to utilize the higher resistance when the diode is reverse biased.

9.3.2 ATG Report

In the ATG all special aspects that are found by the ATG during the generation of the program are listed. Here all messages are allocated precisely to those components for which the generation of the test produced the related messages. At the same point, the warnings that are generated by the ATG in case of components at risk are given.

Example (without warnings):

```
 ATG REPORT
==========


   Created:        2009-08-12 15:00:08
   ATG Version:    ATG 02.83
   EGTSL Version:  EGTSL 02.83


Input:
------
Board Description:  'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example1.BDL'
Physical Layer:     'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example_Physical.ini'


Output:
-------
ICT Program:        'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example1.ict'
ICC Program:        'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example1.icc'
Application Layer:  'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example1_Application.in
Report:             'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example1_Report.txt'



BDL Parser: 0 error(s) and 0 warning(s).


BDL cross check: OK.


>>> Message from TR1                              <<<
TS-PSU not configured in 'physical.ini', simple transistor test inserted


Test coverage report
                    Number  Weight  Coverage
Components             8
Regular tests         7       1      87.5%
Reduced tests         1      0.5      6.25%
Short/Contact test only  0   0.1        0%
Total test coverage                  93.8%
Wiring List info: Nodes mapped to a total number of 7 pin(s).
```

9.3.2.1 Typical Warning Messages

Example:

BDL entry

```
NAME 'R1'
     VALUE 10 kOHM                 Statement with Error
     TOL+ 10%  TOL- 10%
     I_LIM 100.0 MA
     PIN_1 'VCC'
     PIN_2 'MESS1'
```

Error message in the ATG report

```
lexical error (NO BDL keyword) in line 4: k

BDL Parser: 1 error(s) and 0 warning(s).



ERRORS OCCURRED DURING TEST GENERATION:
------------------------------------
ERROR: BDL Parser: syntax errors detected.
```

Error rectification

In BDL it is only allowed to use upper case.

Example:

BDL entry

```
NAME 'R2'
     VALUE 100 OHM
     TOL+ 10%  TOL- 10%
     I_LIM 100.0 MA
     PIN_1 'VCC'
     PIN_3 'GND'                   Statement with Error
```

Error message in the ATG report

```
line 15: syntax error, unexpected PIN_3.

BDL Parser: 1 error(s) and 0 warning(s).



ERRORS OCCURRED DURING TEST GENERATION:
------------------------------------

ERROR: BDL Parser: syntax errors detected.
```

Error rectification

Keyword `PIN_3` is not defined for this component. For this component there is only `PIN_1` and `PIN_2`.

Example:

BDL entry

```
NAME 'R3'
     VALUE 330 OHM
     TOL+ 10%  TOL- 10%
     I_LIM 100.0 UA              Statement with Error
     PIN_1 'MESS1'
     PIN_2 'GND'
```

Error message in the ATG report

```
Error message in the ATG report
line 20: syntax error, unexpected 'U'.


BDL Parser: 1 error(s) and 0 warning(s).



ERRORS OCCURRED DURING TEST GENERATION:
-------------------------------------


ERROR: BDL Parser: syntax errors detected.
```

Error rectification

In the entry `I_LIM` only the unit MA (mA) is allowed, not UA (µA).

9.3.2.2 Alternative Proposals from the ATG

In some cases the ATG generates alternative proposals for measurements, e.g., for guarded resistor measurements or the measurement of very low value resistors. It generates a standard test with higher error limits and, as an alternative, a more exact test with lower error limits.

The following example information relates to Chapter B, "Example 2", on page 300.

The proposals for the individual components are listed in the ATG report.

Example:

Example:

- Guarded measurement of the resistor R1 (10 kΩ)
- Measurement of the low value resistor R4 (10 Ω)

```
BDL Parser: 0 error(s) and 0 warning(s).


BDL cross check: OK.


>>> Message from R1 6-Wire Test Proposal          <<<
Total error for 3-wire mode: 25.52 %
              6-wire mode: 3.1255%


>>> Message from R1                               <<<
Test only possible by exceeding the error limits
Guarding error limiting test quality


>>> Message from R4 4-Wire Test Proposal          <<<
Total error for 2-wire mode: 24,336%
              4-wire mode: 1     %


>>> Message from R4                               <<<
Test only possible by exceeding the error limits
Test limits include system residuals of 2.4 Ohms


Wiring List info: Nodes mapped to a total number of 8 pin(s).
```

Resistor R1	For the measurement of the resistor R1 the ATG proposes two measuring methods. • Guarded 3-wire measurement The standard measurement (3-wire mode) can be performed with three test points (nodes). A disadvantage of the guarded 3-wire measurement is a larger measuring tolerance. The measurement proposed is only possible according to the ATG report if the component tolerances given are exceeded. • Alternative: guarded 6-wire measurement The alternative proposal of a guarded 6-wire measurement has a considerably lower measuring tolerance. However, the guarded 6-wire measurement requires three additional test points (nodes) in the adapter.
Resistor R4	The ATG proposes two measuring methods for the measurement of the resistor R4. • 2-wire measurement The standard measurement (2-wire mode) requires two test points (nodes). A disadvantage of the 2-wire measurement is a larger measuring tolerance. The measurement proposed is only possible according to the ATG report if the component tolerances given are exceeded. • Alternative: 4-wire measurement The alternative proposal of a 4-wire measurement has a significantly lower measuring tolerance. However, the 4-wire measurement requires two additional test points (nodes) in the adapter.

ⓘ During the generation of the ICT program by the ATG, both the standard measurement and the alternative measurement are generated as test steps.

The alternative measurements are marked in the ICT program as *Proposal* variants.

In the Application Layer Configuration File, the additional test points (nodes) required for the alternative measurement are specially marked.

Here it is important that the decision on whether a standard measurement or an alternative measurement (e.g. 2- or 4-wire measurement) is to be performed is made before the adapter is manufactured. In this way additional test points (nodes) required can be taken into account during the manufacture of the adapter.

9.3.2.3 Test Coverage Report

The Test Coverage Report gives an overview of test coverage provided by ATG.

```
Test coverage report
Number   weight   coverage
Components              129
Regular tests          110      1      85.3%
Reduced tests            3    0.5      1.16%
Short/Contact test only 16    0.1      1.24%
Total test coverage                    87.7%
```

Test coverage is calculated by the number of tests generated in reference to the number of components in the BDL file. Tests that are generated are categorized in one of three classes and weighted according to their quality.

Components	The number of components in the BDL file (NODE entries are not counted).
Regular tests	The number of regular tests generated by ATG. Regular tests are rated with a factor of 1.0.
Reduced tests	The number of reduced tests generated by ATG. These include: • 0-Ohm resistors, because a continuity test is inserted instead of a resistor test. • Resistor arrays, if a test was not able to generate all resistors in the array. • Inductors, if a continuity test was generated instead of an impedance test. Diodes and LEDs, if it was not possible to generate any reverse test (current measurement). • Zener diodes, if only a simple diode test could be generated. • Transistors, if only a simple transistor test and no transistor beta test could be generated. Reduced tests are rated with a factor of 0.5.
Short / contact test only	The number of components for which no test could be generated by ATG. Generally this includes **ICs** and **Black Boxes**. The pins of these components are, however, taken into account in the **Contact** and **Short** test. These components are rated with a factor of 0.1.
Total test coverage	Complete test coverage is calculated as a percentage from: $$coverage = \frac{regular + 0.5 * reduced + 0.1 * shortcontact}{components} * 100$$

9.3.3 Application Layer Configuration File

In the Application Layer Configuration File, the specific information for the use of the hardware is compiled by the ATG for the ICT program and the ICC program generated. On the execution of the ICT program or the ICC program, the name of the Application Layer Configuration File generated must be given.

You will find further information on the Application Layer Configuration File in

Example:

```
[ResourceManager]
; general trace settings (normally off)
Trace           = 0
TraceFile       = %GTSLROOT%\resmgr_trace.txt


;--------------------------------------------------------------------------
[LogicalNames]
```

```
ICT   = bench->ICT


;------------------------------------------------------------------------------
[bench->ICT]
Description     = ICT bench generated from ATG
Simulation      = 0
Trace           = 0
ICCDevice       = device->psam
ICTDevice1      = device->psam
ICTDevice2      = device->pict
SwitchDevice1   = device->pmb1
AppChannelTable = io_channel->ICT
AppWiringTable  = io_wiring->ICT


[io_channel->ICT]
GND           = pmb1!P1
INPUT         = pmb1!P2
OUTPUT        = pmb1!P3
TR1.B         = pmb1!P4
TR1.C         = pmb1!P5
TR1.E         = pmb1!P6
VCC           = pmb1!P7


;------------------------------------------------------------------------------


[io_wiring->ICT]
GND           = F1 S15 X10A1
INPUT         = F1 S15 X10A2
OUTPUT        = F1 S15 X10A3
TR1.B         = F1 S15 X10A4
TR1.C         = F1 S15 X10A5
TR1.E         = F1 S15 X10A6
VCC           = F1 S15 X10A7


;------------------------------------------------------------------------------
```

In the `[io_channel->ICT]` section of the Application Layer Configuration File, the
test points (nodes) are allocated to the I/O channels on the R&S TS-PMB Matrix Mod-
ule B. The information on which matrix modules are available in the test system is read
from the Physical Layer Configuration File.

9.3.4 Adapter Manufacture

The information necessary for the manufacture of the adapter is in the Application
Layer Configuration File generated (`[io_channel->ICT]` and `[io_wiring->ICT]`)
(see Chapter 9.3.3, "Application Layer Configuration File", on page 192).

9.3.5 ICC Program

The Automatic Test Generator ATG generates an ICC program for the IC-Check test of a circuit board.

An IC-Check program file is a test file with the extension `.icc`. It can be created or edited using any text editor. The logical structure of the ICC program file is made up of information blocks on ICs to be tested. Each information block begins with the name of the block in square brackets. All the subsequent lines begin with a keyword and either describe the measurement configuration to be applied to test the individual pins of the IC or the names of the pins and reference pins to be tested. All pins of the BDL files having the letter sequence "GND", "VSS" or "VCC" (case insensitive) are listed in the IC-Check program file as reference pins.

Refer to *chapter 8 R&S IC-Check Test Programs* of the document *Software Description IC-Check* for a detailed description of the contents and usage of an ICC program file.

10 Test Methods

10.1 Test Hardware
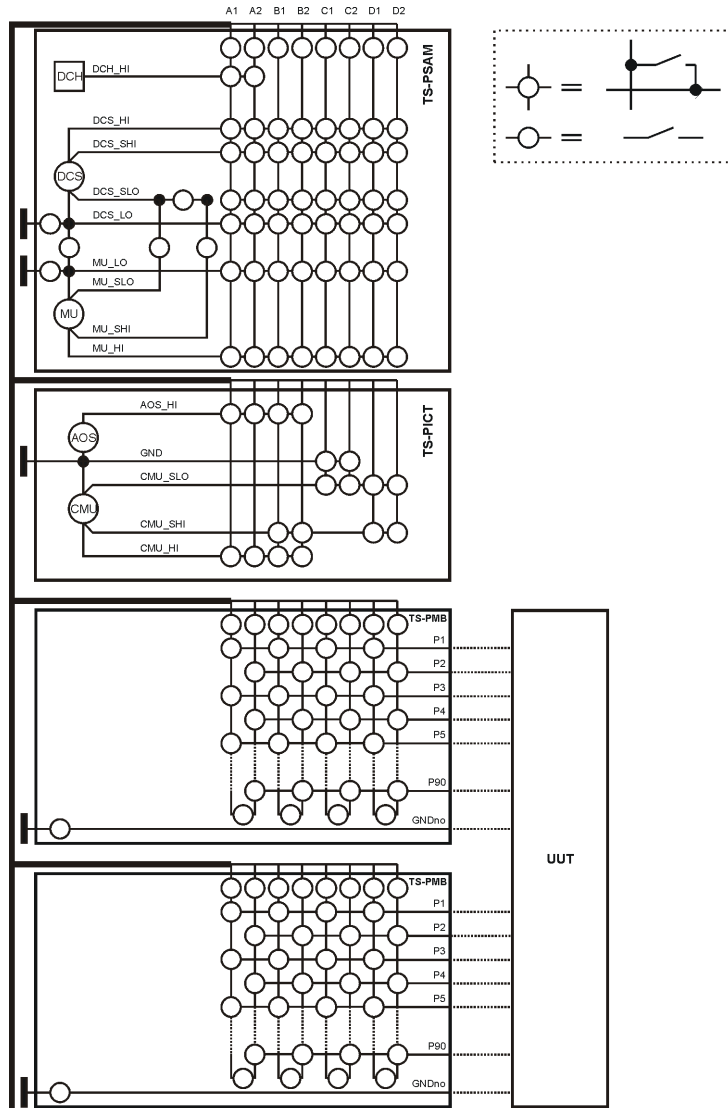
Figure 10-1: Test hardware

The following test hardware is required to perform the individual tests (see Figure 10-1):

- **R&S TS-PSAM Source and Measurement Module**

Table 10-1: DC stimulation source (DCS)

max. voltage	5 V
max. current	100 mA
The current limit is adjustable	

Table 10-2: Voltage and current measurement unit (MU)

Smallest voltage measuring range DC	10 mV
Smallest voltage measuring range AC	20 mV rms
Smallest current measuring range DC	1 µA
Smallest current measuring range AC	100 µA

Table 10-3: Discharge unit DCH

max. voltage	30 V
max. current	400 mA

- **R&S TS-PICT In-Circuit Test Module**
  The R&S TS-PICT In-Circuit Test Module is only required for the diode test, the impedance test and the guarded resistor test.

Table 10-4: AC voltage source (AOS)

max. voltage AC (rms)	1 V at 100 Hz, 1 kHz, 10 kHz
max. voltage DC	5 V
max. current DC	50 mA

Table 10-5: Current Measuring Unit (CMU)

Current measuring range AC	5 µA to 250 mA

- **R&S TS-PSU Power Supply/Load Module**
  R&S TS-PSU12 Power Supply/Load Module 12V
  The R&S TS-PSU Power Supply/Load Module / R&S TS-PSU Power Supply/Load Module 12V is only required for the zener diode test and the transistor beta test.

max. voltage TS-PSU	50 V (single)
	100 V (cascaded)
max. voltage TS-PSU12	12 V (single)
	24 V (cascaded)
max. current	100 mA

- **R&S TS-PMB Matrix Module B**
  - Connection of the individual measuring pins (nodes)

The measuring and stimulation hardware are connected to the Unit Under Test (UUT) using the R&S TS-PMB and the analog bus.

ⓘ   The exact function and the technical data on the measuring and stimulation hardware are described in detail in the user manuals and the data sheets for the individual modules.

10.2   Ground Wiring

During the in-circuit tests, the UUT (Unit Under test) must be floating, i.e. it must not be connected with the ground of the test system.

The correct ground wiring is shown in Figure 10-1. The ground of the UUT is connected with the GNDno connectors of the matrix card. This can be connected through a relay with the system ground GND. During the in-circuit tests, these relays must be opened. For the function test, they can be closed.

10.3   Contact Test

ⓘ   Starting with GTSL 3.30, no GTSL license is required.

License required:	R&S TS-LEGT or R&S TS-LEG2

Test hardware	Module
DCS	R&S TS-PSAM
MU	R&S TS-PSAM
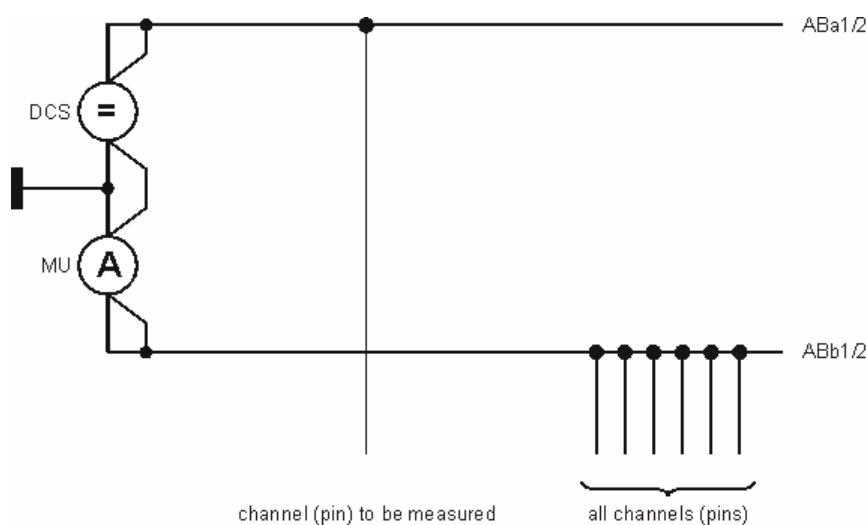
Figure 10-2: Connection for contact test

During the contact test it is to be checked whether there is a contact between all channels (pins) on the list and the unit under test. For this purpose, in principle the resistance between a channel (pin) to be tested and all other channels (pins) is measured.

The contact test is intended to ensure that, at the start of a test program, any errors produced subsequently are not due to poor contacts. It is thus reasonable to stop the test program after contact errors so that the subsequent tests such as the short-circuit test, continuity test and the component test are not performed.

The maximum permissible resistance or the minimum current necessary can be adjusted in the contact test. If the measured resistance is above the threshold value, it is assumed that there is no contact.

The voltage for the measurement is selected such that any diode paths present are forward biased (e.g. U_{set} = 2.5 V). If no current flow can be detected during the first measurement, then the poles of the source are reversed so that any diode paths present are forward biased. During this process the current must be limited such that the components in the circuit are not destroyed (e.g. I_{max} = 50 µA).

The smallest measurable resistance is thus given by U_{set} / I_{max}. For smaller resistances, the DC source reaches the current limit and the resistance cannot be determined from the current measurement only.

Case 1: low resistance connection

The DC source reaches the current limit. The resistance cannot be determined because the voltage of the source is unknown. The current measured corresponds to the current limit. However, the resistance is certainly smaller than U_{set} / I_{max}. (2.5 V / 50 µA = 50 kΩ).

Case 2: high resistance connection or open circuit

The DC source is not at the current limit. The resistance can be determined as follows:

$R = U_{set} / I_{meas}$

In the report on the contact test, the pins without contact are listed. All pins are tested, however a maximum of 10 pins without contact are listed.

10.4 Continuity Test

Starting with GTSL 3.30, no GTSL license is required.

License required:	R&S TS-LEGT or R&S TS-LEG2

(i) The connection for the continuity test is made in a similar manner to the connection for the contact test (see Figure 10-2). However, only two pins (pin pair) are ever connected to the analog bus.

With the aid of the continuity test it is checked whether there is continuity between a list of nodes (pins).

There is continuity when the resistance between two nodes (pins) is less than a defined value.

For the continuity test the voltage is selected such that diode paths are reverse biased (e.g. U_{set} = 0.2 V). The current limit is set to the maximum possible value. In this way capacitors in the circuit are charged more quickly. During the charging process, a large current flows than could also be interpreted as continuity. If there is no continuity, then after the charging time the current will drop to an acceptable, stable value. If there is continuity, the DC source is either at the current limit and supplies the programmed current, or a current limited by the system residuals (approx. 1.2 Ω) flows.

If the DC source is not at the current limit, the resistance of the continuity can be determined as follows:

$R = U_{set} / I_{meas} - R_{resid}$

(i) The low resistance continuities or networks are listed in the report on the continuity test.

Example for report entry:

P1 P2/P3 ...	between P2 and P3 there is no continuity

10.5 Diode Test

(i) Starting with GTSL 3.30, no GTSL license is required.

License required:	R&S TS-LEGT

Test hardware	Module
DCS	R&S TS-PSAM
MU	R&S TS-PSAM
CMU	R&S TS-PICT
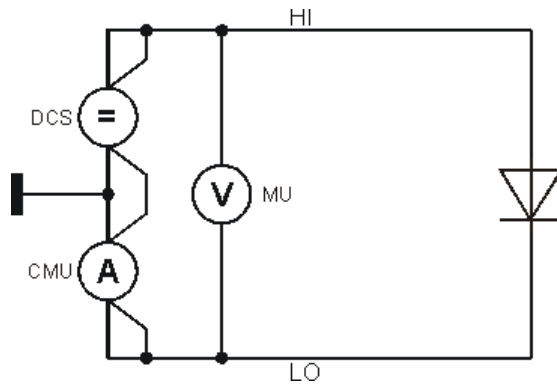
Figure 10-3: Diode test (forward bias voltage + reverse bias current)
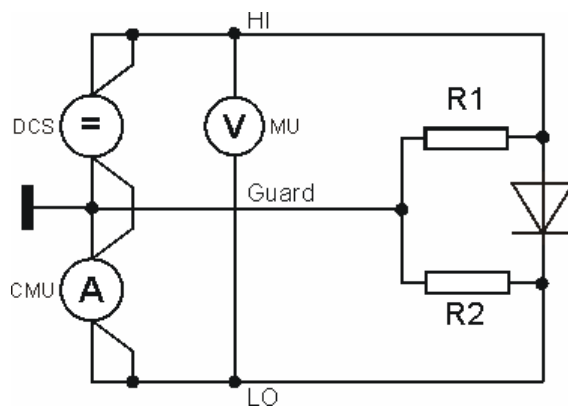
Figure 10-4: Diode test with guarding (forward bias voltage + reverse bias current)

During the diode test, the following tests are performed:

- Forward bias voltage (knee voltage) test
- Reverse bias current test

The same connection between measuring and stimulation hardware and the component to be tested is used for both tests. Only the settings for the test hardware are changed. This measuring process also applies for the diode test with guarding.

10.6 Discharging Capacitors

Starting with GTSL 3.30, no GTSL license is required.

License required:	R&S TS-LEGT or R&S TS-LEG2

Test hardware	Module
DCH	R&S TS-PSAM
DCS	R&S TS-PSAM
MU	R&S TS-PSAM
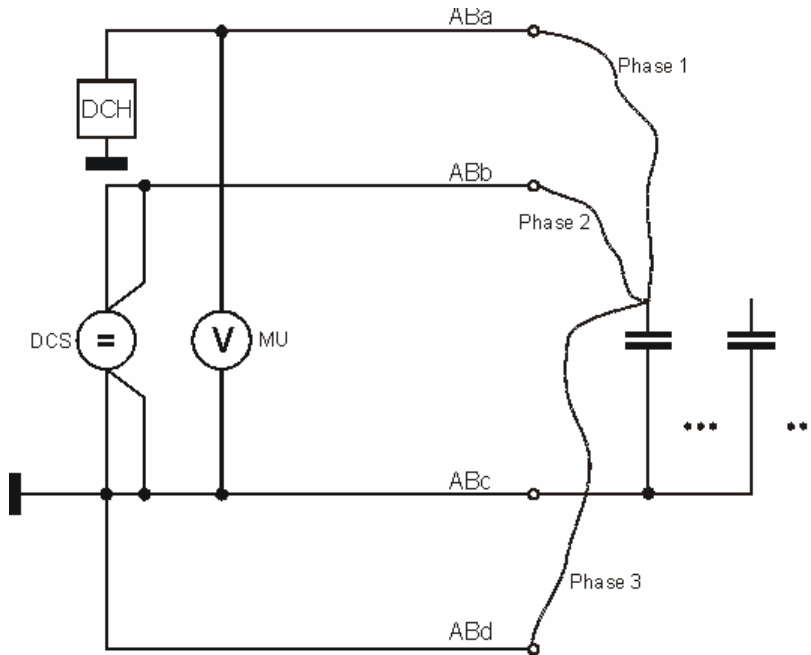
Figure 10-5: Discharge test

The capacitors are discharged in several phases. The main discharge is performed using the discharge circuit (DCH) on the R&S TS-PSAM.

Phase 1	The node / capacitor to be discharged (*test pin*) is connected using the analog bus A (ABa). The node / capacitor is discharged via the discharge circuit (DCH). The voltage at ABa is now measured cyclically. When the measured voltage drops below a defined value, phase 1 is complete.
Phase 2	The node / capacitor to be discharged(*test pin*) is switched from analog bus A (ABa) to analog bus B (ABb) and thus to the DC source (DCS). The discharge is now performed using the DCS at 100 mA. Phase 2 is complete when the DCS is no longer at the current limit.
Phase 3	The node / capacitor to be discharged (*test pin*) is switched to analog bus D (ABd) and thus shorted to the *reference pin*.During this phase the capacitor discharges via the cable resistances on the switching matrix.

If the maximum duration for the discharging of all nodes / capacitors (*test pins*) is exceeded, the test pins that have not yet been discharged are listed in the report. The

pin on which the measuring time expired during discharging and for which the discharge process was not completed is also listed.

10.7 FET Test

ⓘ Starting with GTSL 3.30, no GTSL license is required.

License required:	R&S TS-LEGT

Test hardware	Module
DCS	TS-PSAM
VMU	TS-PSAM
AOS	TS-PICT
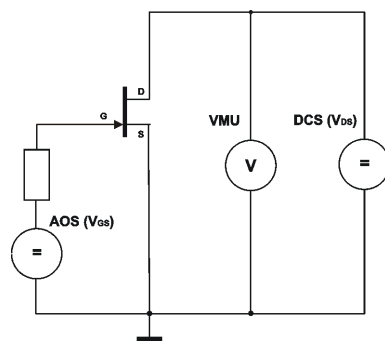
Figure 10-6: FET test

The AOS voltage source generates a voltage between the gate and source of a field-effect transistor. The DCS voltage source generates a voltage between the drain and the source. The value of the drain-source voltage is measured with the VMU voltmeter.

The gate-source voltage is first selected such that the drain-source channel of the FET is blocked (FET off). The set current limiting of the DCS voltage source does not respond. The DCS is in the constant-voltage mode. Voltmeter VMU is used to measure the voltage value set at the DCS.

In a second measurement, the voltage generated by the AOS between the gate and the source is selected such that the drain-source channel of the FET becomes conductive (FET on). The DCS voltage source switches to voltage limiting, i.e. the voltage of the DCS is reduced to the extent that exactly the current flows that is set via the voltage limiting value. Voltmeter VMU will now measure voltage that is considerably lower than the voltage set at the DCS.

10.8   Impedance Test

The impedance test is used for components that cannot be measured using DC. These include capacitors and inductors. Using the AOS (AC-Offset Source) a sinusoidal voltage is applied, the current is measured using the CMU (Current Measurement Unit). At the same time the voltage applied to the measured object is measured using the MU (Measurement Unit). The complex resistance (= impedance) is determined from the two signal curves measured. The measured impedance can be represented as magnitude and phase, e.g. IMP-MAG and IMP-PHASE. It is also possible to represent the impedance as the real and imaginary parts, e.g. RES-SER, REAC-SER.

This impedance can be represented by a serial or parallel equivalent circuit. For a fixed frequency, both the serial and parallel equivalent circuit can be used. The actual component L or C can be calculated from the frequency (e.g. RES-PAR, CAP-PAR). Which equivalent circuit is more relevant to the practical situation depends on the specific case.
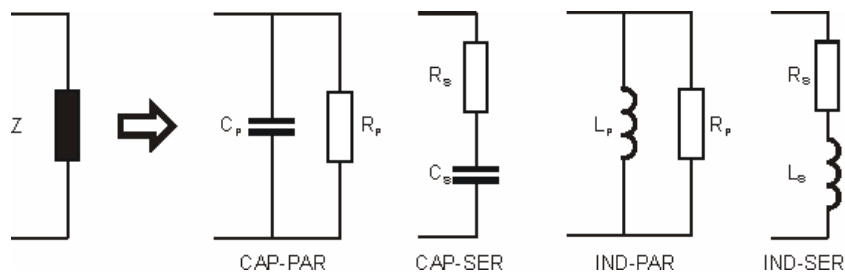
Figure 10-7: Equivalent circuit diagrams for determination of measured value

The values for the individual components vary depending on equivalent circuit. In practice the circuits drawn are often used. Here C_P and C_S (L_P and L_S) are relatively equal, while R_P has a very high value, and R_S has a very low value. Nevertheless, the related circuits have the same impedance.

10.8.1   2- and 4-wire measurement

ⓘ   Starting with GTSL 3.30, no GTSL license is required.

License required:	R&S TS-LEGT

Test hardware	Module
AOS	R&S TS-PICT
CMU	R&S TS-PICT
MU	R&S TS-PSAM
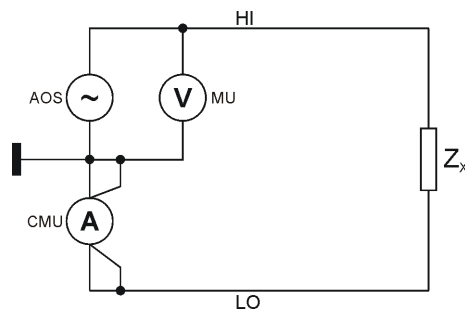
Figure 10-8: 2-wire impedance measurement (mode V)
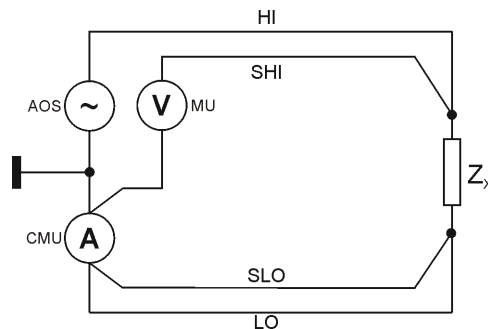
Figure 10-9: 4-wire impedance measurement (mode VS)

On the R&S EGTSLin conjunction with the Test System Versatile Platform R&S CompactTSVP (R&S PowerTSVP), all impedances are measured in mode V or VS, this means that a voltage is applied and the current is measured. To determine in particular the exact phase, the voltage must also be measured at the same time. In mode V the measuring instruments are connected directly together and two measuring wires fed from here to the unit under test (see Figure 10-8). During this simple 2-wire impedance measurement, all impedances in series are included as errors. You can compensate for this situation by connecting the voltmeter to the unit under test using separate wires and measuring the voltages directly (see Figure 10-9). For this purpose, four wires to the unit under test and the four nails in the bed of nails adapter are necessary (4-wire impedance measurement, also called Kelvin measurement).

10.8.2 Guarded Measurement

Starting with GTSL 3.30, no GTSL license is required.

License required:	R&S TS-LEGT

Test hardware	Module
AOS	R&S TS-PICT
CMU	R&S TS-PICT
MU	R&S TS-PSAM
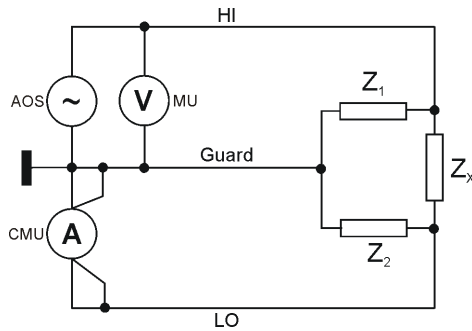
Figure 10-10: Guarded 3-wire impedance measurement
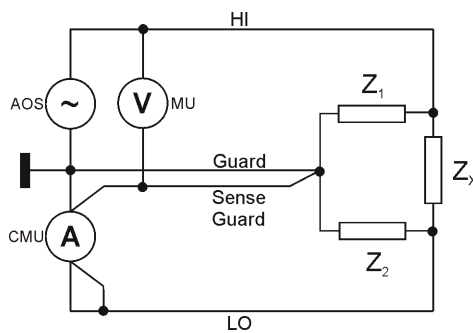
Figure 10-11: Guarded 4-wire impedance measurement
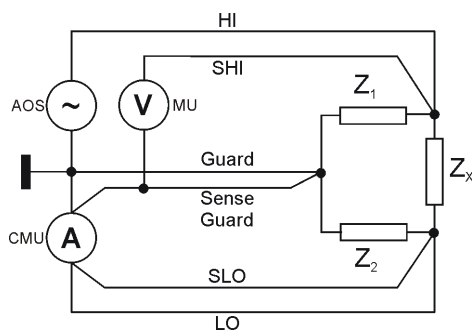
Figure 10-12: Guarded 6-wire impedance measurement

If there are several parallel branches on the unit under test, the parallel components can be electrically isolated ("guarding"). This is possible when each parallel path comprises at least two serial components. All these components can be combined to the components Z_1 and Z_2 (see Figure 10-10). The CMU ensures electronically that at Z_2 the voltage 0 V is present and in this way no current flows through Z_2. In this way the CMU can measure exactly the current that also flows through Z_X. The current through

Z_1 does not interfere with the measurement. This method is called guarding or 3-wire measurement.

In reality the wires are, however, not ideal such that voltage drops are produced on the wires due to the resistances of the wires ("residuals"). You can compensate for this problem by connecting separate sensor wires directly to the unit under test and measuring the voltage there. (Guarded 6-wire impedance measurement, see Figure 10-12). However, 6 nails must also be provided and the tester must also provide six pins. The main error on this measurement is produced by the guard path. For this reason there is a simplified guard measurement that only senses the guard branch (guarded 4-wire impedance measurement). This compromise only requires four nails (see Figure 10-11). The measuring accuracy achievable is between the guarded 3-wire impedance measurement and the guarded 6-wire impedance measurement.

The guarded 4-wire impedance measurement must not be confused with the 4-wire impedance measurement (mode VS).

The lower the value of the impedances Z_1 or Z_2 referred to Z_X, the more difficult the measurement becomes. Guard ratios of up to 1:1000 can be measured ($Z_1 : Z_X$ or $Z_2 : Z_X$ = 1 : 1000). However, with these guard ratios, a considerable compromise on measuring accuracy must be made.

During the circuit analysis by the Automatic Test Generator ATG such difficult measurements are detected and marked in the generation report with messages and alternative proposals.

In practice the guarded 6-wire impedance measurement is used where high accuracy is required. The sensing requires

- More nails.
- More wiring in the adapter.
- A larger test setup (number of pins).

Overall the measurement becomes slower, the measurement throughput thus reduces.

10.8.3 Correct Phase Impedance Measurement

The correct phase impedance measurement enables components directly in parallel to also be measured within limits
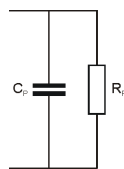
The accuracy is best when the contributions for the real and reactive elements are of the same order of magnitude. At a low frequency R_P can be measured very precisely, as in this case the reactive element of C_P increases. R_P can be determined even better with a pure DC measurement.

10.8.4 System Residuals

A basic problem is the differing system residuals depending on the system configuration. Every additional module connected to the analog bus changes the residuals by a few pF. A special correction method is used so that test programs run the same on different systems. Using the program ICT Correction (see also Chapter 15.1, "System Residual Compensation", on page 245), all impedance measurements on a "typical pin" can be corrected. In this way different configurations and module fits can be corrected.

10.8.5 Fixture Residuals

Besides residuals given by the actual circuit, additional impedances are inserted by the measurement adapter. This is inevitable and can only be compensated but not eliminated. If a test is done on different measurement adapter, it is necessary to distinguishing the residual impedances and the fixture residuals. Only with separated residuals, it is possible to define the fixture residuals for the different adapter. The actual compensation is done in two steps.

1. An overall correction value is applied by debugging the ICT program. Those correction values include both residuals since the system residual cannot be measured on its own.

2. The fixture residual must be determined by running an open adapter measurement. With the measured values, the difference to the values in step 1 are calculated and preserved as correction values.
   It is not necessary to scope with the complete complexity of the residual compensation. If only one adapter shall be used, the correction step is sufficient to compensate any disturbances of the actual measurement setup. To enable the fixture residual compensation, an option in the APPLICATION.INI file must be set, see Chapter 3.2.3, "Entries in APPLICATION.INI", on page 22. Per default, when this option is not active, the EGTSL works in a legacy mode. This is to maintain compatibility of stored ICT programs with older EGTSL versions (≤3.33).

ⓘ Programs that are enhanced by the compensation information cannot be loaded within an EGTSL Software with Version ≤3.33.

If such a file must be loaded again in an EGTSL Software with Version ≤3.33, the ICT program can be transformed back. For this reason, load the ICT again in an EGTSL Version ≥3.34 with deactivated compensation support, and save the file directly afterwards. This will result in the same file, except the compensation information is deleted. If there are already compensation values not equal 0.0, the software automatically adds those compensation values to the correction value and deletes the compensation tag afterwards.

See also Chapter 15.2, "Adapter Compensation", on page 250.

10.8.6 Measuring Small Capacitors

A particular challenge is the measurement of small capacitors, particularly in the range of 10 pF and smaller. Here there are capacitances (typically 60 pF to 90 pF) on the switching panel and analog bus in parallel that introduce errors directly into the measurement. In addition, there is scatter on the capacitances between two pins due to the switching panel and the adapter wiring. Only the scatter between switching panel pins is included in the measurement. These errors can be determined by reference measurements and taken into account in the measured result.

See also Chapter 15.2, "Adapter Compensation", on page 250.

10.8.7 Measurement of Polarized Capacitors/Electrolytic Capacitors

Electrolytic capacitors only have their defined capacitance when a DC test voltage with the correct polarity is applied. For this reason electrolytic capacitors are stimulated with an AC voltage with an additional DC offset. Pole reversal up 0.5 V is allowed.

10.9 Resistor Test

10.9.1 2-wire Measurement

(i) Starting with GTSL 3.30, no GTSL license is required.

License required:	R&S TS-LEGT or R&S TS-LEG2

Test hardware	Module
DCS	R&S TS-PSAM
MU	R&S TS-PSAM
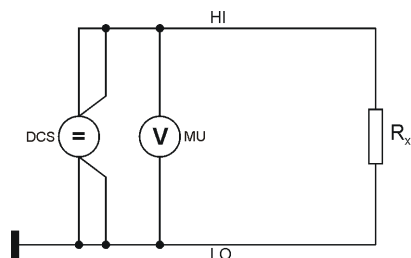
Figure 10-13: 2-wire resistor measurement mode C

Mode C = application of current with voltage measurement

Mode C is intended for the measurement of low value resistors (<10 Ohm) with low accuracy requirements.

(i) To keep the errors low, low value resistors should always be measured using the 4-wire resistor measurement mode C.
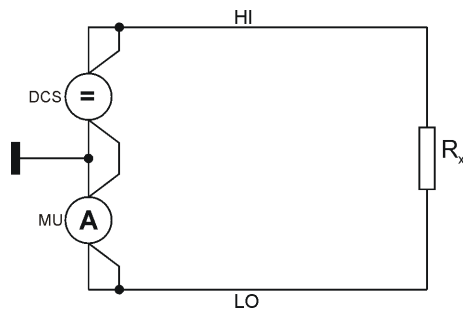
Figure 10-14: 2-wire resistor measurement mode V

Mode V = application of voltage with current measurement

The 2-wire resistor measurement mode V is the standard measuring method for high value resistors. The 2-wire resistor measurement mode V is also used for the measurement of resistors in a range from 100 Ω to 1 MΩ if medium measuring accuracy is sufficient. The measurement is faster than the 4-wire measurement mode VS.

Due to the application of voltage, mode V has the advantage that capacitances on the Unit Under Test (UUT) are charged quickly and the measuring speed is thus increased. Due to the defined voltage that always remains the same, parallel diode paths and ICs are not forward biased or are only evenly slightly forward biased. This increases the reproducibility of the measurement.

10.9.2 4-wire Measurement

(i) Starting with GTSL 3.30, no GTSL license is required.

License required:	R&S TS-LEGT or R&S TS-LEG2

Test hardware	Module
DCS	R&S TS-PSAM
MU	R&S TS-PSAM
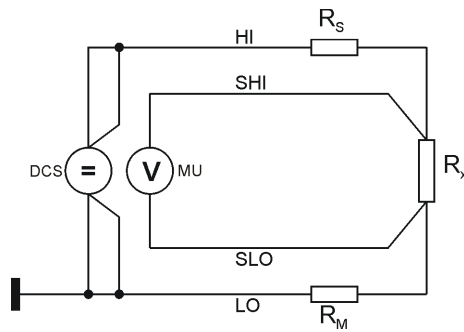
Figure 10-15: 4-wire resistance measurement mode CS

Mode C = application of current with voltage measurement

The 4-wire resistance measurement mode CS is intended for resistors in the range 0.01 Ω to 10 Ω.

By measuring the voltage directly at the component to be tested, voltage drops on the wires do not affect the result of the measurement.
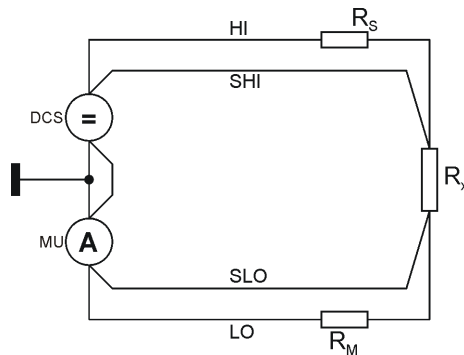
Figure 10-16: 4-wire resistor measurement mode VS

Mode V = application of voltage with current measurement

The 4-wire resistor measurement mode VS is intended for resistors in the range 10 Ω to 10 kΩ.

Scattering on the switching paths can be almost completely compensated by sensing. The measuring speed is however lower than for measurement without sensing.

10.9.3 Guarded Measurement

Starting with GTSL 3.30, no GTSL license is required.

License required:	R&S TS-LEGT

Test hardware	Module
DCS	R&S TS-PSAM
CMU	R&S TS-PICT

ⓘ All guarded measurement take place in mode V (application of voltage with current measurement).

The same requirements as for the 2-wire and the 4-wire measurement apply.
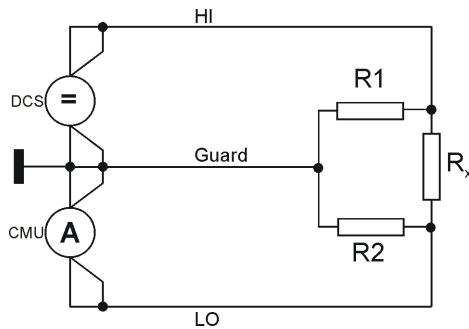
Figure 10-17: Guarded 3-wire resistor measurement
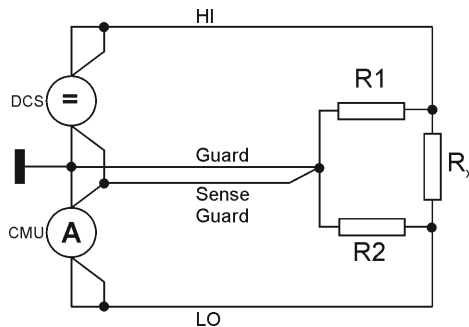
Figure 10-18: Guarded 4-wire resistor measurement
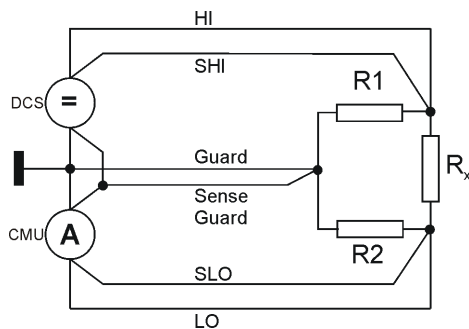
Figure 10-19: Guarded 6-wire resistor measurement

The mode V method (application of voltage with current measurement) has the same advantages for guarded resistor measurements (speed and reproducibility) as for

unguarded resistor measurements. For this reason, mode C (application of current with voltage measurement) is not used for guarded resistor measurements.

The system residuals (series resistors and series inductors) can be compensated by sensing. The error due to the guard path is in general an order of magnitude higher that the error on the stimulation and measuring path.

The most precise measuring method "guarded 6-wire resistor measurement" also requires the most effort for test nails in the test adapter and tester setup. The "guarded 4-wire resistor measurement" is a compromise here, as only the particularly high guard error is removed. However, only 4 nails are needed for this measurement.

10.10 Short-circuit Test

(i) Starting with GTSL 3.30, no GTSL license is required.

License required:	R&S TS-LEGT or R&S TS-LEG2

(i) The connection of the short-circuit test is performed in exactly the same manner as for the connection of the contact test (see Figure 10-2)

With the aid of the short-circuit test (short) it is intended to detect short-circuits between two nodes (pins) on an assembly.

There is a short-circuit when a resistance that is less than a defined resistance threshold is measured between a channel (pin) and all other channels (pins).

During this test the voltage is selected such that diode paths are not forward biased (e.g. U_{set} = 0.2 V). The current limit is set to the maximum possible value. In this way capacitors in the circuit are charged more quickly. During the charging process, a large current flows that could also be interpreted as a short-circuit. If there is no short-circuit, then after the charging time the current will drop to an acceptable, stable value. If there is a short-circuit, the DC source is either at the current limit and supplies the current programmed, or a current limited by the system residuals flows.

To perform this measurement at maximum speed, it is necessary to monitor the transient process (timing). As long as the DC source is at the current limit, or the current flow is changing, capacitances are still being charged or there is a short-circuit. Only if the flow of current drops to a value under the threshold after a certain amount of time is there no short-circuit. The response time is essentially dependent on the capacitances present, the wire resistances, the parallel resistance and the programmed current limit.

The short-circuit test implemented in R&S EGTSL features high test speed and high reliability at the same time. This is achieved by using a two-stage test method.

In the first stage, each pin (test point, node) is checked against all other pins. However due to the parallel switching used during this process, a large number of pseudo short-

circuits (e.g. due to capacitors) are detected that are not actually "real" short-circuits. R&S EGTSL therefore only considers these "suspicious pins".

In the second stage these "suspicious pins" are tested again. In this stage the "timing" parameters for the short-circuit test are applied. Finally, only real short-circuits (connections or networks) are listed in the report

10.11 Transistor Test

ⓘ Starting with GTSL 3.30, no GTSL license is required.

License required:	R&S TS-LEGT

Test hardware	Module
DCS	R&S TS-PSAM
MU	R&S TS-PSAM
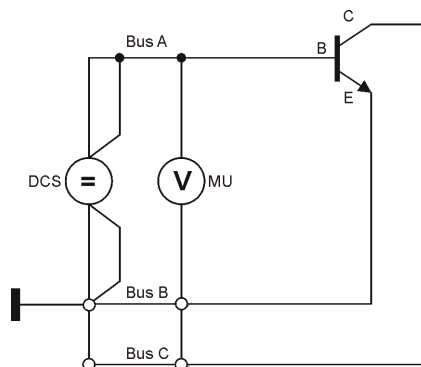
Figure 10-20: Transistor test (voltage measurement)

During the transistor test the forward bias voltage (knee voltage) for the base-emitter diode and the base-collector diode are measured. Guarding is not performed.

Using the transistor test, pins that are not soldered and incorrectly positioned components (e.g. PNP instead of NPN) can be found. A beta test is not performed.

10.12 Transistor Beta

ⓘ Starting with GTSL 3.30, no GTSL license is required.

License required:	R&S TS-LEGT

Test hardware	Module
CH2	R&S TS-PSU / R&S TS-PSU12 (channel 2)
DCS	R&S TS-PSAM
VMU	R&S TS-PSAM
CMU	R&S TS-PICT
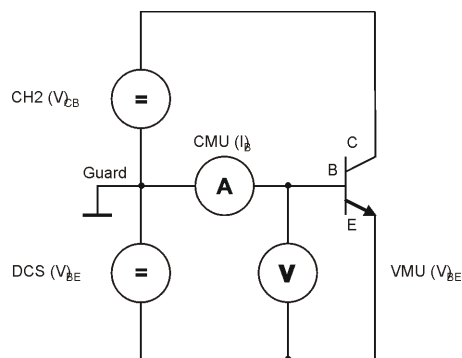
Figure 10-21: Transistor Beta test

The voltage source R&S TS-PSU / R&S TS-PSU12 generates a voltage between the base and collector of the transistor. Current limiting must be set high enough so that the source can provide sufficient current for the transistor and any branches wired in parallel that may be present.

The voltage source DCS generates a voltage between the base and emitter of the transistor. Current limiting must be adjusted so that the source is working in constant current mode.

The forward bias voltage (knee voltage) over the base emitter diode is measured in the first measurement (voltage V_{BE}, see Figure 10-21).

A constant current I_E is impressed in the emitter branch with the DCS for the measurement of the current gain (Beta β, see Figure 10-21). At the same time the current I_B is measured in the base branch. Then the emitter current is changed by a small amount and a second measurement of the base current is performed. The dynamic current gain Beta β is calculated from these measurements as follows:

$$\beta = \frac{\Delta I_C}{\Delta I_B} = \frac{\Delta I_E - \Delta I_B}{\Delta I_B}$$

The current gain Beta β is calculated from the ratio of the change in current in the collector circuit to the change in current in the base circuit. The collector current can be expressed as the difference between the emitter current and the base current.

In the event of a guarded measurement, all guard points are connected with the ground reference point.

The measurement for PNP transistors runs in the same manner, with the difference that the polarity of the two voltage sources is reversed.

10.13 Zener Diode

ⓘ Starting with GTSL 3.30, no GTSL license is required.

License required:	R&S TS-LEGT

Test hardware	Module
CH1/CH2	R&S TS-PSU / R&S TS-PSU12
MU	R&S TS-PSAM
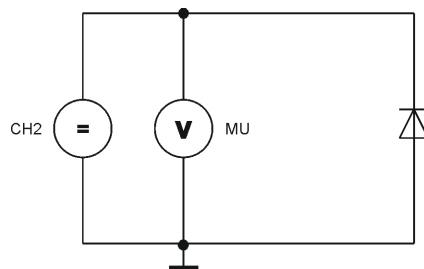
Figure 10-22: Zener Diode test
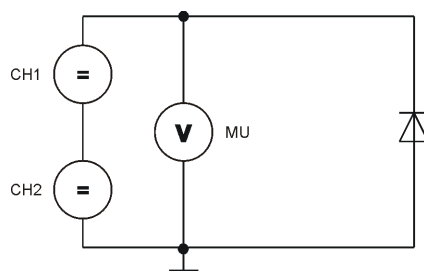
Figure 10-23: Zener Diode test (> 50 V / 12 V, cascaded)

During the zener diode test, the voltage source R&S TS-PSU / R&S TS-PSU12 generates a voltage in the reverse direction of the diode which is somewhat above the Zener voltage. Current limiting must be adjusted so that the voltage source is working in constant current mode. Now the working voltage (zener voltage) can be adjusted by the zener diode and measured with a voltmeter.

An individual channel of the voltage source R&S TS-PSU can produce a maximum of 50 V (R&S TS-PSU: 12 V). To be able to test Zener diodes with higher voltages, you have to cascade the channels of the module. This cascading is not possible on the module itself. It must be provided in the test adapter.

1. Turn off the R&S TS-PSU module.

2. **WARNING!** Risk of electric shock. Serial wiring allows you to generate voltages of up to 100 V. Voltages higher than 30 V RMS, or 42 V peak, or 60 V DC are regarded as hazardous contact voltages. Direct contact with them can cause serious injuries.
   Make sure that only electrically skilled persons use the products for measurements on hazardous contact voltages. For more information on how to avoid serious injuries, follow the safety instructions for working with high voltages. The safety instructions are part of the delivery of the R&S TSVP base unit.

   In addition, voltages higher than 120 V can damage or destroy the module. Floating channels must therefore never exceed a voltage of 120 V against ground. This rule applies to each individual connection pin and is even more important for cascaded sources (for example when they are switched together in the adapter).

   Connect outputs CH2_HI4 and CH1_LO4 on connector X10 on the module to each other (see Figure 10-23 and Figure 10-24).
   Note that you must not use the multiplexer outputs represented by hatched lines in Figure 10-24 if you cascade channels for a Zener diode test.

3. **NOTICE!** Risk of instrument damage. Serial wiring can also lead to high voltages and currents running through the system. High voltages or currents can destroy the module or the unit under test.
   To avoid damage to the module, make sure that the current stays below the values the module and unit under test can handle.

   Set the current limit to an appropriate value.

4. Enable the R&S TS-PSU output.
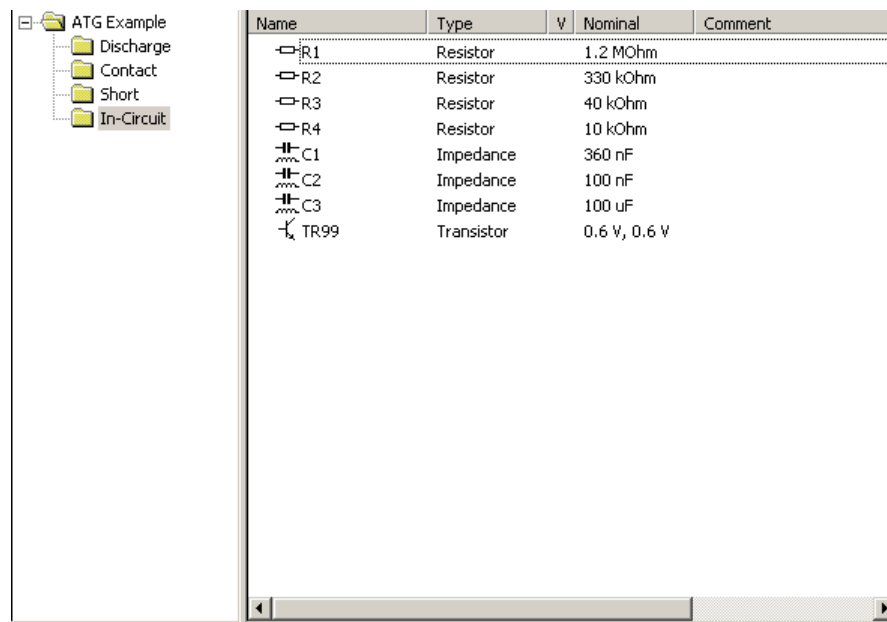
	A	B	C
1	LABA1	GND	LABA2
2	LABB1	GND	LABB2
3	LABC1	GND	LABC2
4	LABD1	GND	LABD2
5			
6	CH1_HI1	CH1_HI1	CH1_HI1
7	CH1_LO1	CH1_LO1	CH1_LO1
8	CH1_HI2	CH1_HI2	CH1_HI2
9	CH1_LO2	CH1_LO2	CH1_LO2
10	CH1_HI3	CH1_HI3	CH1_HI3
11	CH1_LO3	CH1_LO3	CH1_LO3
12	CH1_HI4	CH1_HI4	CH1_HI4
13	CH1_LO4	CH1_LO4	CH1_LO4
14	CH1_SHI1		CH1_SLO1
15	CH1_SHI2	CH1_MHI	CH1_SLO2
16	CH1_SHI3	CH1_MLO	CH1_SLO3
17	CH1_SHI4		CH1_SLO4
18	CH2_HI1	CH2_HI1	CH2_HI1
19	CH2_LO1	CH2_LO1	CH2_LO1
20	CH2_HI2	CH2_HI2	CH2_HI2
21	CH2_LO2	CH2_LO2	CH2_LO2
22	CH2_HI3	CH2_HI3	CH2_HI3
23	CH2_LO3	CH2_LO3	CH2_LO3
24	CH2_HI4	CH2_HI4	CH2_HI4
25	CH2_LO4	CH2_LO4	CH2_LO4
26	CH2_SHI1		CH2_SLO1
27	CH2_SHI2	CH2_MHI	CH2_SLO2
28	CH2_SHI3	CH2_MLO	CH2_SLO3
29	CH2_SHI4		CH2_SLO4
30			
31	XTI1	XTI2	GND
32	XTO1	XTO2	CHA_GND

X10 C 13 (connects to C13 CH1_LO4)

X10 C 24 (connects to C24 CH2_HI4)

Figure 10-24: Zener Diode Test, Adapter Cabling for Voltage > 5 V

11 Creating Test Programs

11.1 Program Groups

It is practical to divide the ICT program into several program groups. This is done auto-matically when an ICT program is created with the Automatic Test Generator ATG (see Chapter 9.3.1, "ICT Program", on page 183).

Figure 11-1: Program sub-window

When an ICT program is started with a function call of the ICT test library (e.g. from C-Program in production mode) the executable program group can be transferred as a parameter. Only the selected program group is executed. If, during the running of the program group, one or more errors occur (e.g. if the discharge or the contact for the unit under testing have failed), the testing of the unit can be immediately aborted. The ICT program does not have to run to the end. This way, unnecessary testing time can be avoided.

Through the **<Step Properties>**, the **Fail Count** function can be activated for the pro-gram group (see Chapter 5.6.2.3, "Menu Command <Step Properties>", on page 121).

	The call is made either through the menu command **<Edit><Step Properties>**, the key combination *[Alt +Enter]*, the button **Step Properties** or the context menu.

When the **Fail Count** function is activated, the erroneous test steps in a program group are counted. If the given number of erroneous test steps is reached, none of the sub-

sequent test steps in the program group are executed. The **Fail Count** function is only active when an ICT program is started through the functions of the ICT test library.

An example of combining individual test steps into a program group and working with the **Fail Count** function is the testing of resistor arrays. A test step is created for each individual resistor of the resistor array. The individual test steps are combined into a program group. The **Fail Count** of the program group is set to the value "1". If a resistor of the resistor array tests as "Failed", the testing of the other resistors in the program group is cancelled and the measured result of the entire program group is set to "Failed".

A further example of the use of the **Fail Count** function is the activation at the highest program group, i.e. for the entire ICT program. For the ICT program in Figure 11-1, this is the program group `ATG Example`. As soon as an error occurs in one of the program groups lower in the hierarchy, the testing of the further program groups is cancelled and the ICT program is terminated with the result "Failed" for the unit under test.

11.2 Variants

11.2.1 Use

By using variants, units under test which show only small deviations from one another can be tested with the same ICT program. Using a single ICT program has the following advantages:

- One uniform ICT program for the same test steps or same test parts.
- Changes need only be carried out in one ICT program.
- Only one ICT program needs to be tested (Debugging).

Test steps (or groups) with variant definitions are used in the following cases in the ICT program:

- A component is assembled in the original version but not in the variant XYZ. The test step is only executed in the original version.
- A component has a different value range in the original version than in the variant XYZ. Thus, other measuring tolerances will result for the test step. In an extreme case, another test method is defined for the test step in the variant XYZ.
- When using special test strategies, it may be practical to test a component with various test methods and measuring tolerances.
- Using your own variant definition, test steps can be activated and deactivated while debugging the ICT program.
- Variants can be defined for comments and non-executable test steps.
- During the automatic generation of an ICT program with the Automatic Test Generator ATG, the alternative proposals for measurements created by the ATG are identified as variants (variant name: *Proposal*). Also see Chapter 9.3.2.2, "Alternative Proposals from the ATG", on page 189.

11.2.2 Definition

Before a variant can be used, it must be defined in the **<Program Properties>** (see Chapter 5.6.1.6, "Menu Command <Program Properties>", on page 112). The call is made using the menu command **<File><Program Properties>**. A name and a short description are entered for the variant. The defined variants are valid for the entire ICT program.

11.2.3 Assignment

A variant can be assigned to a test step or a group. The assignment is made through **Variant Selection** in **<Step Properties>** (see Chapter 5.6.2.3, "Menu Command <Step Properties>", on page 121).

	The call is made through either the menu command **<Edit><Step Properties>**, the key combination **[Alt +Enter]**, the **Step Properties** button or the context menu.

Test steps or groups to which a variant is assigned are identified in the debug mode (R&S EGTSL IDE) in the program sub-window with a "**V**" in the variant column.

Figure 11-2: Variant Identification

In the program sub-window, the deactivated variants are shown as crossed through.

Figure 11-3: Deactivated variants

11.2.4 Execution

When an ICT program is started through the functions of the ICT test library (e.g. from C-Program in production mode), the executable variant can be transferred as a parameter. The ICT program is executed with the selected variant.

When an ICT program is started in debug mode (R&S EGTSL IDE), the executable variant is displayed in the **Debug** sub-window in the **Variant** menu. Also see Chapter 5.4.6, "Debug Subwindow", on page 46.

11.3 Limit Files

Through the use of limit files, it is possible to separate the limit values from the ICT program. The limit files can be stored and processed in a central place. By calling different limit files with different limit values in an ICT program, a more detailed report on the unit under test is possible (e.g. narrower measuring tolerances).

By using different limit values for an ICT program, variant control is also possible. Thus, identical units under test which differ from one another simply in the tolerances of the assembled components, can be tested with the same ICT program.

Before an ICT program is started through the functions of the ICT test library, one or more limit files can be loaded (e.g. from C-Program in production mode). The ICT program is executed with the loaded limit values.

In the limit files, only the limit values without units of measurement are stored. The units of measurement are stored in the ICT program.

When the limit files are loaded, only the limit values which are entered in the limit files are temporarily changed in the ICT program. Test steps for which no limit values are entered in the limit files remain unchanged.

Further information for loading, import and export of the limit values in debug mode (R&S EGTSL IDE), and the format of the limit files can be seen in Chapter 5.6.1.7, "Menu Command <Limits>", on page 114.

11.4 Multiple Panel Testing

If components are manufactured in multiple panels, one ICT program is created for all panels. This process does not differ from the creation of an ICT program for a single unit (e.g. creation by the Automatic Test Generator ATG). It is not practical to create a separate ICT program for each unit of the multiple panel. Using only one ICT program a has the following advantages:

● One uniform ICT program for all panels.
● Changes need only be carried out in one ICT program.
● Only one ICT program needs to be tested (Debugging).

Because, as a rule, the design and the assembly of the individual panels is identical, only the individual different contacts need to be adjusted.

The control of the ICT program for the individual panels is done through bench entries in the Application Layer Configuration File (see example in section Chapter 11.4.1, "Example of an Application Layer Configuration File for Multiple Uses", on page 222). Each panel is assigned its own bench entry with its own I/O channel list. In the I/O channel lists, the different I/O channels of the matrix cards R&S TS-PMB are assigned to the identical test points (nodes). When possible, a unique matrix card R&S TS-PMB should be used for the wiring of each individual panel.

When the ICT program is loaded, the bench which is to be executed must be entered. This also applies to a call through the functions of the ICT test library (e.g. from C-Program in production mode), and to a call through the R&S EGTSLloader (R&S EGTSL IDE in debug mode). Thus, all panels can be tested by calling the same ICT program with different benches assigned.

11.4.1 Example of an Application Layer Configuration File for Multiple Uses

When an ICT program is created by the Automatic Test Generator ATG, only one bench with one I/O channel list is generated. For use during a multiple panel test, the Application Layer Configuration File must be processed in a text editor. The bench generated by the ATG must be copied and the individual entries adjusted.

```
[ResourceManager]
; general trace settings (normally off)
Trace          = 0
TraceFile      = %GTSLROOT%\resmgr_trace.txt


;----------------------------------------------------------------------------
[LogicalNames]
ICT_1   = bench->ICT_Panel_1
ICT_2   = bench->ICT_Panel_2
ICT_3   = bench->ICT_Panel_3
ICT_4   = bench->ICT_Panel_4


;----------------------------------------------------------------------------
[bench->ICT_Panel_1]
Description     = ICT bench Panel 1 (Simulation)
Simulation      = 1
Trace           = 0
ICTDevice1      = device->psam
ICTDevice2      = device->pict
SwitchDevice1   = device->pmb1
AppChannelTable = io_channel->ICT_Panel_1


[io_channel->ICT_Panel_1]
GND          = pmb1!P1
INPUT        = pmb1!P2
```

```
OUTPUT        = pmb1!P3
TR1.B         = pmb1!P4
TR1.C         = pmb1!P5
TR1.E         = pmb1!P6
VCC           = pmb1!P7


;-------------------------------------------------------------------------------
[bench->ICT_Panel_2]
Description     = ICT bench Panel 2 (Simulation)
Simulation      = 1
Trace           = 0
ICTDevice1      = device->psam
ICTDevice2      = device->pict
SwitchDevice1   = device->pmb2
AppChannelTable = io_channel->ICT_Panel_2

[io_channel->ICT_Panel_2]
GND           = pmb2!P1
INPUT         = pmb2!P2
OUTPUT        = pmb2!P3
TR1.B         = pmb2!P4
TR1.C         = pmb2!P5
TR1.E         = pmb2!P6
VCC           = pmb2!P7


;-------------------------------------------------------------------------------
[bench->ICT_Panel_3]
Description     = ICT bench Panel 3 (Simulation)
Simulation      = 1
Trace           = 0
ICTDevice1      = device->psam
ICTDevice2      = device->pict
SwitchDevice1   = device->pmb3
AppChannelTable = io_channel->ICT_Panel_3

[io_channel->ICT_Panel_3]
GND           = pmb3!P1
INPUT         = pmb3!P2
OUTPUT        = pmb3!P3
TR1.B         = pmb3!P4
TR1.C         = pmb3!P5
TR1.E         = pmb3!P6
VCC           = pmb3!P7


;-------------------------------------------------------------------------------
[bench->ICT_Panel_4]
Description     = ICT bench Panel 4 (Simulation)
Simulation      = 1
Trace           = 0
ICTDevice1      = device->psam
```

```
ICTDevice2      = device->pict
SwitchDevice1   = device->pmb4
AppChannelTable = io_channel->ICT_Panel_4


[io_channel->ICT_Panel_4]
GND             = pmb4!P1
INPUT           = pmb4!P2
OUTPUT          = pmb4!P3
TR1.B           = pmb4!P4
TR1.C           = pmb4!P5
TR1.E           = pmb4!P6
VCC             = pmb4!P7


;-----------------------------------------------------------------------------
```

12 Debugger

12.1 Overview

The debugging of an ICT program in the Enhanced Generic Test Software Library R&S EGTSL is done in the R&S EGTSL user interface (*R&S EGTSL IDE*). In the debug mode of the *R&S EGTSL IDE*

- the ICT program can be executed test step by test step.
- the measured results can be analyzed.
- corrections can be made in the ICT program.

The ICT program is immediately ready to be executed after the changes, without having to be compiled and restarted.

The following actions are possible in debug mode:

- starting and stopping of the ICT program
- stepwise execution of the ICT program
- continuation the ICT program at another position
- repeated execution of test steps in a loop
- entering and immediately executing test steps
- setting of stop conditions and breakpoints
- changing of measurement parameters
- analysis of measured values

(i)   In the following, the R&S EGTSL- user interface (*R&S EGTSL IDE*) is referred to as the debugger.

12.2 Starting and Terminating the Debugger

For the debugging of an ICT program, the R&S EGTSL user interface (debugger) must be started. The debugger can be started in two ways.

1. Start the debugger from a test sequence through a function call from the ICT Test Library (see Chapter 13.1, "Starting Using a Function Call from the ICT Test Library", on page 235)

2. Start the debugger through the loader (see Chapter 13.2, "Starting Using the R&S EGTSL Loader", on page 238)

When the debugger is started from a test sequence, the ICT program is automatically opened with the appropriate bench. When the debugger is started through the R&S EGTSL loader, the ICT program must be manually opened with the menu command

"File -> Open". When the ICT file is opened, the appropriate bench for the execution of ICT program must be entered.
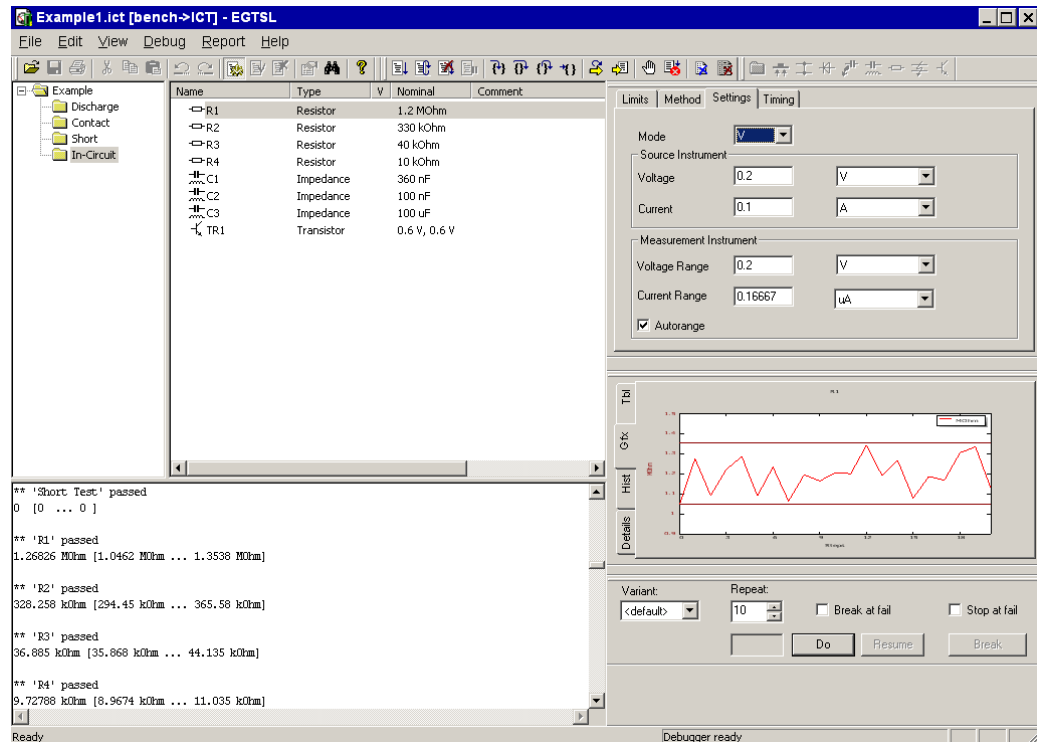
Figure 12-1: Debugger

ⓘ The individual screens and functions of the debugger (*R&S EGTSL IDE*) are described in Chapter 5, "User Interface (R&S EGTSL IDE)", on page 37.

The debugger is terminated with the menu command **<Exit>** (also see Chapter 5.6.1.10, "Menu Command <Exit>", on page 118).

12.3 Debug Status

The status of debugger is shown by messages in the status bar.

Debugger ready	The debugger is activated. One or more test steps, program groups or the entire ICT program can be executed (see Chapter 12.4, "Using the Debugger", on page 227section).
Debugger deactivated	The debugger is deactivated. The ICT program or test steps cannot be executed.
ICT is running	One or more test steps, program groups or the entire ICT program are executed with the given number of repetitions.

Total execution time for the last executed debugger command is shown in the status bar.

12.4 Using the Debugger

The debugger is operated through the commands from the main menu item, or through buttons, function keys or key combinations.

If the debugger has stopped the ICT program, a yellow arrow on the left side points to the test step to be executed next. The execution of the program will be continued at this point.

The debugger stops the ICT program, if

- the command has been executed, for example, a single step.
- a breakpoint is reached.
- a stop condition is fulfilled.
- a runtime error has occurred

Go *[F5]*	Executes the ICT program once from the test step marked with the yellow arrow (current debugger execution point) to the end. If the debugger is deactivated, the ICT program is executed from start to finish.
Restart *[Ctrl+Shift+F5]*	With the debugger activated, executes the ICT program from the start to the finish.
Terminate *[Shift+F5]*	Stops the execution of the ICT program. The marking with the yellow arrow (current debugger execution point) is removed. The debugger is deactivated.
Break	Interrupts the execution of the ICT program. The test step at which the interruption takes place is marked with the yellow arrow (current debugger execution point) and displayed. Using the **Go** command the execution of the ICT program can be continued.
Repeat *[F4]*	With the debugger activated, starts the execution of the test steps marked (test steps with a color background) with the number of repetitions given.
Resume *[Shift+F4]*	Continues the execution of the marked test steps (test steps with a color background) following an interruption using the **Break** button.
Step Into *[F11]*	Executes the ICT program from the test step marked with the yellow arrow (current execution point of the debugger) in single steps. If the test step is a group, automatically each test step of the group will be executed in single steps.

Step Over *[F10]*	Executes the ICT program from the test step marked with the yellow arrow (current execution point of the debugger) in single steps. If the test step is a group, all test steps in the group are performed automatically.
Step Out *[Shift+F11]*	If the test step marked with the yellow arrow (current execution point of the debugger) is in a group, the group is left with this button. The test steps that follow in the group are performed automatically. If the test step marked with the yellow arrow (current execution point of the debugger) is at the top program level, all following test steps including groups are executed.
Run to Cursor *[Ctrl+F10]*	Executes the ICT program from the test step marked with the yellow arrow (current execution point of the debugger) to the test step marked with the cursor. The test step marked by the position of the cursor is marked as the next test step that can be executed using the yellow arrow.
Set Next Step [*Ctrl+Shift+F10]*	Marks the position of the cursor as the test step that is to be executed next. A yellow arrow is displayed on the left beside the test step marked. The debugger is activated.
Show Next Step *[Alt+F10]*	Jumps to the current debugger execution position (test step marked with the yellow arrow). The test step is the next that is executed.
Toggle Breakpoint *[F9]*	Sets a breakpoint at the cursor position. The breakpoint is deactivated when you click the button again. The third time you click the button, the breakpoint is deleted (see also Chapter 5.6.2.4, "Menu Point <Breakpoints>", on page 124).
Stop At Fail	With this function activated, the ICT program execution is stopped when the test step result was "Fail". The program stops at the erroneous test step. The **Stop at fail** function applies to all debug commands with the exception of the **Repeat** function.
Break At Fail	With this function activated, the execution of the ICT program (test steps marked) is interrupted when the test step result was "Fail". The program stops at the erroneous test step. The **Break at fail** function applies only to the **Repeat** function.

12.5 Breakpoints

The execution of the program in the debugger can be interrupted at predetermined points (test steps or program groups) by using the breakpoints. The debugger stops the ICT program at this point, before it executes the test step which has been assigned an activated breakpoint.

Toggle Breakpoint *[F9]*	Breakpoints are set, deactivated or deleted through the menu command , the function key *[F9]* or the *Toggle Breakpoint* button.

Breakpoints are displayed on the left in the program sub-window next to the marked test step or on the left next to the marked program group. The debugger does not stop at deactivated breakpoints.

	activated breakpoint
	deactivated breakpoint

Using the menu command or the key combination *[Alt+F9]*, a list of all breakpoints set in the ICT program can be displayed. In the editing window, the breakpoints can be individually or totally deleted. In addition, you can jump to a selected breakpoint. Also see Chapter 5.6.2.4, "Menu Point <Breakpoints>", on page 124.

12.6 Stop Conditions

The execution of the program in the debugger can be interrupted when specified conditions occur.

Stop At Fail	With this function activated, the ICT program execution is stopped when the test step result was "Fail". The program stops at the erroneous test step. The **Stop at fail** function applies to all debug commands with the exception of the **Repeat** function.
Break At Fail	With this function activated, the execution of the ICT program (test steps marked) is interrupted when the test step result was "Fail". The program stops at the erroneous test step. The **Break at fail** function applies only to the **Repeat** function.

12.7 Typical Debugging Procedure

The optimum procedure for testing an ICT program (debugging) is different for each unit under test and thus for each ICT program. The debugging of a new ICT program should be done with a reference unit under test (Reference UUT).

The following steps should only be used as a reference for a debugging procedure. The steps described apply to an ICT program created "by hand" as well as to one generated by the Automatic Test Generator ATG.

1. Open a new ICT program with corresponding bench in the debugger.

2. Check the proposals created as variants by the Automatic Test Generator ATG and select the best variants for the test procedure as the standard test steps (also see Chapter 9.3.2.2, "Alternative Proposals from the ATG", on page 189).

3. Activate the function **Stop At Fail** and start the ICT program with the function **GO** (menu item, button or function key *[F5]*).

4. The debugger interrupts the execution of the ICT program at the **Failed** test step. The test step is shown by the yellow arrow (current execution position of the debugger).

5. Repeat the **Failed** test step with the function **Repeat** (menu item, button or function key *[F4]*) (more than once, if necessary).

6. Check the measured results with the information from the **Report** sub-window and the **Results Tbl, Results Gfx, Results Hist** and **Results Details** sub-windows.

7. Based on the information from the **Report** sub-window and the **Results** sub-window, adjust the measurement parameters of the test step.

8. Start the further execution of the ICT program with the function **GO** (menu item, button or function key *[F5]*).

9. Repeat steps 4. to 8. until the ICT program runs without errors.

10. Start the ICT program with the function **GO** several times (menu item, button or function key *[F5]*) and let it run through (without errors).

11. Check the measured results for the entire ICT program with the information from the **Report** sub-window and the **Results Tbl, Results Gfx, Results Hist** and **Results Details** sub-windows (see section 12.9).

12. If the ICT program with the reference unit under test (Reference UUT) functions without error, the step must be repeated with other units under test (UUT) until the ICT program also runs without errors with these units under test.

13. If the ICT program can be run for several units under test (UUT) without the error, a measuring time optimization can be carried out as a last step (see Chapter 12.8, "Measuring Time Optimization", on page 230).

12.8 Measuring Time Optimization

Through an exact analysis of the measured results in the **Results Details** sub-window, information on the measuring time (duration) and the status of the measuring hardware used for the measurement can be ascertained.

A measuring time optimization can be achieved by:

● Optimum selection of the measurement range of the measuring hardware used at each test step.

● Adjustment of the timing settings (Delay, Max.Wait Interval, Max.Wait accuracy, Average, Sample Interval) at each test step.

- Selection of the appropriate testing method (2-wire measurement, 4-wire measurement, guarding) for each test step.

12.9 Interpretation of the Results Sub-window

The graphical display of the measured results of the individual test steps in the Results Gfx and Results Hist sub-windows gives important information for optimization of the individual measuring procedures. A few typical examples are shown below.

12.9.1 Results Gfx Sub-Window

Example:
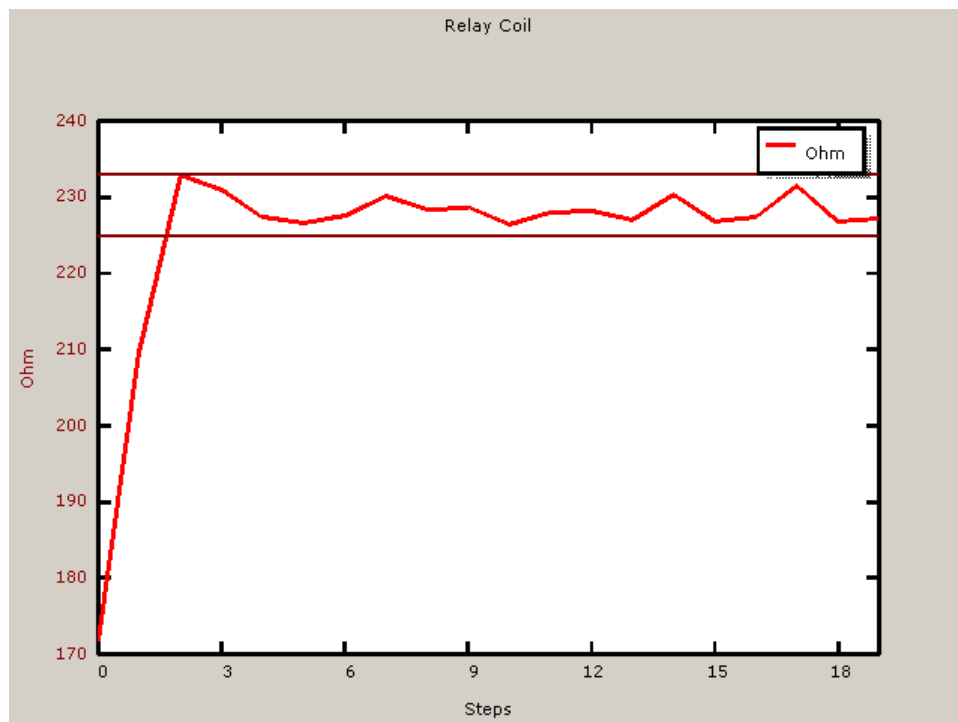
Figure 12-2: Measurements with Transient Process

The graph in Figure 12-2 shows a measurement with transient process. The measured results are stable and lie within the given measuring tolerances only after the third run of the test step. For the optimization of this test step, the auto-delay technique would be useful (see Chapter 5.5.12.1, "Timing", on page 103). By using this, you can ensure that a correct measured result is obtained even for the first run of the test step.

Example:
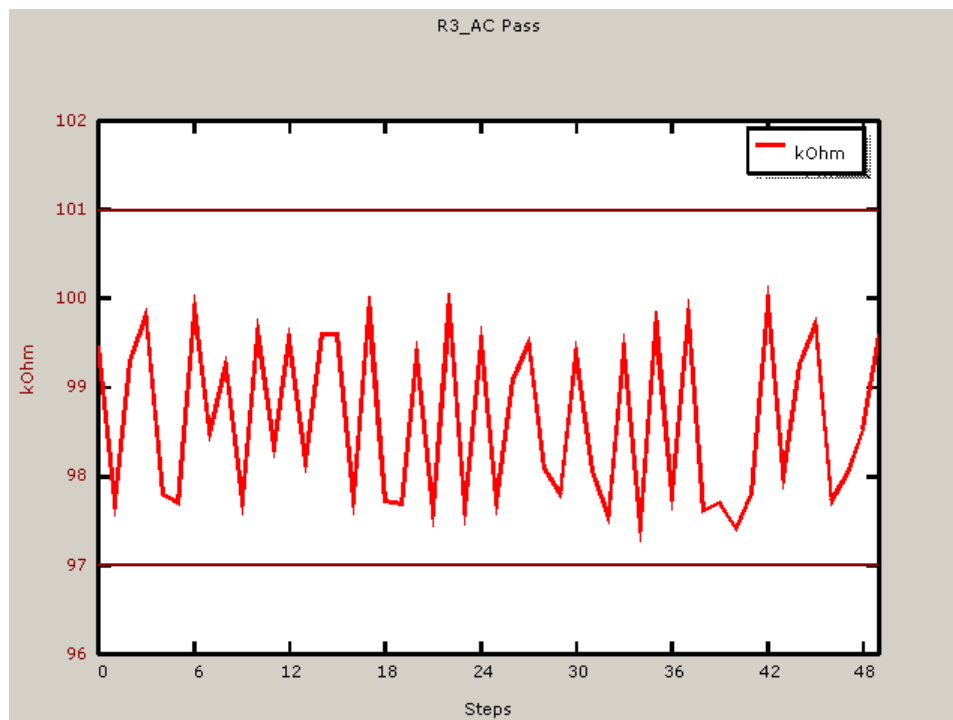
Figure 12-3: Measurement with superimposed interference

The graph in Figure 12-3 displays a measurement with superimposed interference signal. Through an exact circuit analysis, the source of interference can be determined and eliminated with appropriate measures (e.g. another test method).

12.9.2   Results Hist Sub-Window

Example:
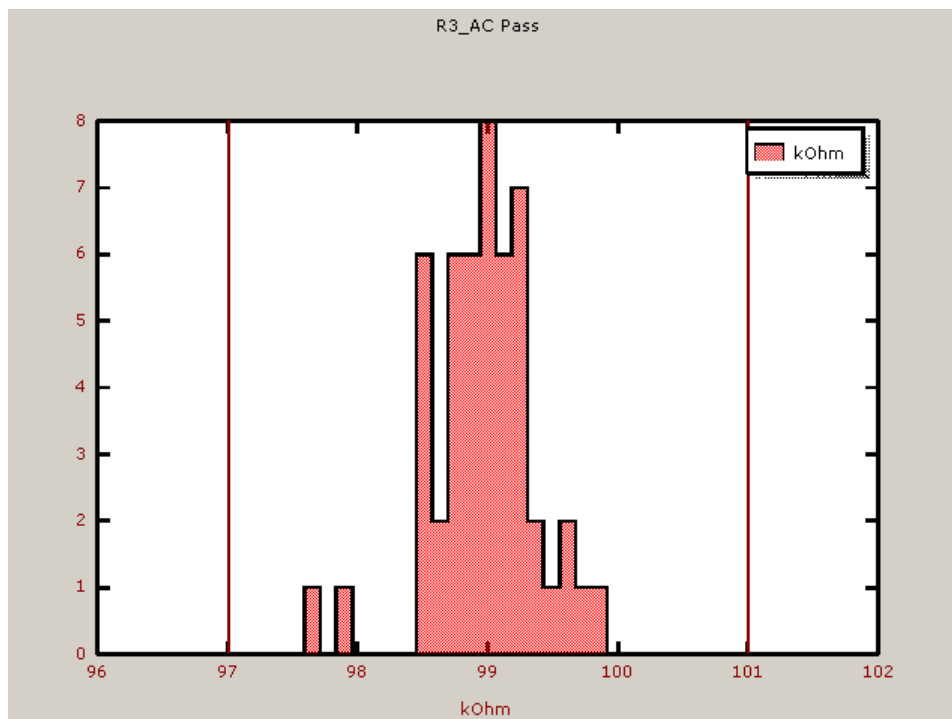
Figure 12-4: Measurements with wide scatter
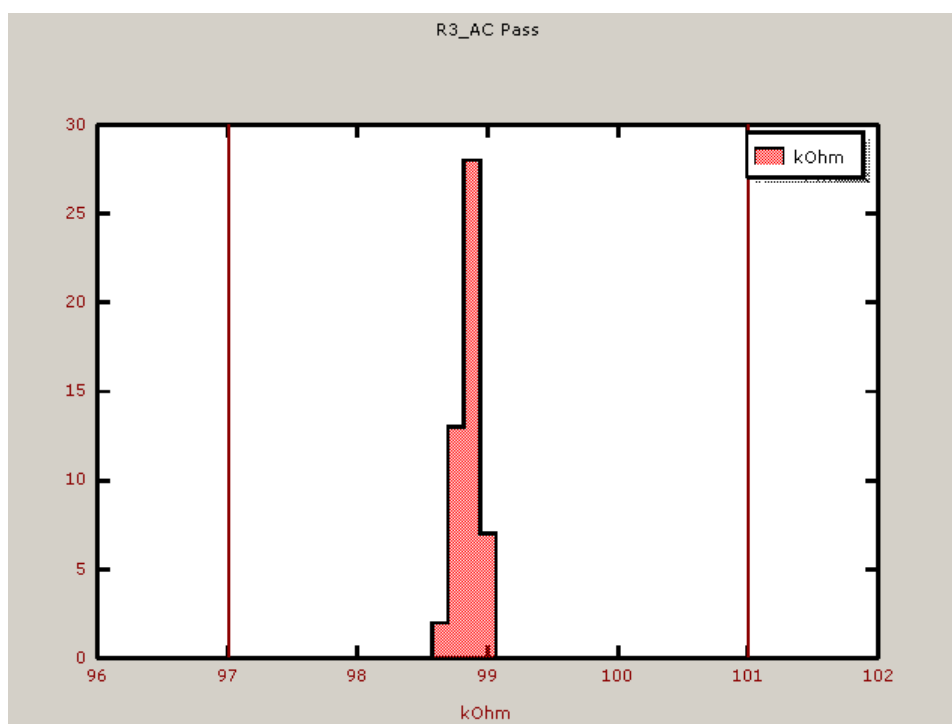
Figure 12-5: Measurements with narrow scatter

The graphs in Figure 12-4 and Figure 12-5 show the distribution of the measured values of a test step. The scatter of the measured results for the measurements in Figure 12-4 is too wide. The cause of this wide scatter could be superimposed interference or non-optimum timing settings. The measurement can be improved with **Max. Wait** or **Average** settings (see Chapter 5.5.12.1, "Timing", on page 103).

13 Running R&S EGTSL IDE

The Enhanced Generic Test Software Library R&S EGTSL can be executed in two ways:

1. By opening a function from the ICT test library.

2. Using the R&S EGTSL Loader.

ⓘ Starting with GTSL 3.30, no GTSL license is required.

13.1 Starting Using a Function Call from the ICT Test Library
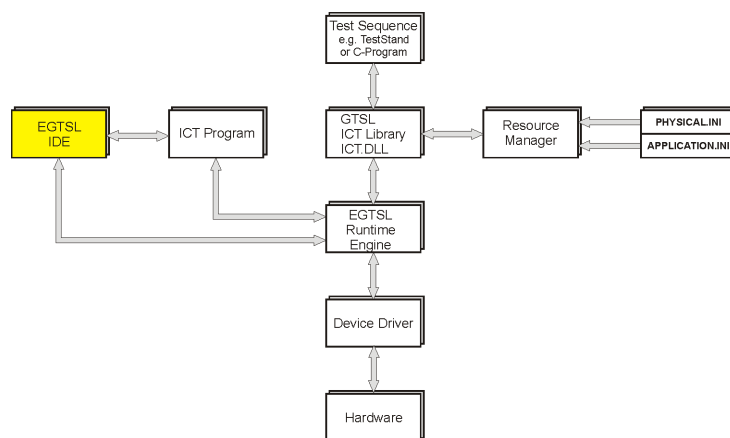
Figure 13-1: Opening R&S EGTSL IDE using test library (software structure)

Functions from the ICT test library are opened from the test sequence (e.g. test sequence in test stand or a dedicated C program). The test hardware available is managed using the resource manager with the aid of the configuration files (`PHYSICAL.INI` and `APPLICATION.INI`). The R&S EGTSL runtime engine now opens the R&S EGTSL user interface R&S EGTSL) and the corresponding ICT program. The ICT program opened can now be debugged in the R&S EGTSL user interface (R&S EGTSL). Using the R&S EGTSL runtime engine the ICT program can be executed on the test hardware directly from the R&S EGTSL user interface.

13.1.1 Example

In the following example, functions from the ICT test library are opened from the C program. Among other aspects, the R&S EGTSL user interface (R&S EGTSL IDE) can be opened.

```c
#include <cviauto.h>
#include <ansi_c.h>
#include <userint.h>
#include <cvirte.h>
#include "ict.h"

// defines the drive and path for the example files
#define PATH "C:\\Program Files\\Rohde&Schwarz\\GTSL\\EGTSL\\example\\"

int main (int argc, char *argv[])
{
  short errorOccurred = FALSE;
  long  errorCode;
  char  errorMessage[GTSL_ERROR_BUFFER_SIZE] = "";
  char  userMessage[1024];
  long  resourceId = -1;
  long  programId  = -1;
  long  compileErrors;
  long  failCount;
  int   response = 0;

  // ActiveX must be initialized if the ICT library is used
  CA_InitActiveXThreadStyleForCurrentThread (0, COINIT_APARTMENTTHREADED);

  // load the configuration files
  if ( ! errorOccurred )
  {
    RESMGR_Setup (0, PATH "example_physical.ini",
                     PATH "example1_application.ini",
                     &errorOccurred, &errorCode, errorMessage);
  }

  // load the ICT library
  if ( ! errorOccurred )
  {
    ICT_Setup (0, "bench->ICT",
                  &resourceId,
                  &errorOccurred, &errorCode, errorMessage);
  }


  // load the ICT program
  if ( ! errorOccurred )
  {
    ICT_Load_Program (0, resourceId,
                         PATH "example1.ict",
                         "bench->ICT",
                         &programId,
                         &compileErrors,
                         &errorOccurred, &errorCode, errorMessage);
```

```
            }


    do
    {
      // start the program
      if ( ! errorOccurred )
      {
        if ( 1 == ConfirmPopup ( "EGTSL", "Start in debug mode?" ) )
        {
          // debug the program with EGTSL IDE
          ICT_Debug_Program (0, resourceId,
                                programId,
                                "",          // no variant specified
                                "",          // no subtreee, run complete program
                                1,           // stop at start of program
                                1,           // stop at end of program
                                0,           // don't stop at fail
                                &failCount,
                                &errorOccurred, &errorCode, errorMessage);
        }
        else
        {
          // run the program without EGTSL IDE
          ICT_Run_Program (0, resourceId,
                              programId,
                              "",            // no variant specified
                              "",            // no subtree, run complete program
                              &failCount,
                              &errorOccurred, &errorCode, errorMessage);
        }
      }

      // write the report to a text file
      if ( ! errorOccurred )
      {
        ICT_Write_Report (0, resourceId,
                            programId,
                            "ictreport.txt",
                            0,            // 0 = overwrite, 1 = append
                            &errorOccurred, &errorCode, errorMessage);
      }

      // ask the user to run the program again
      if ( ! errorOccurred )
      {
        sprintf ( userMessage, "%d tests failed\n\nTry again?", failCount );
        response = ConfirmPopup ("EGTSL", userMessage );
      }
      else
```

```
    {
      break;
    }


  } while (1 == response);
  // error handling
  if ( errorOccurred )
  {
    sprintf ( userMessage, "Error Code %d:\n\n%s", errorCode, errorMessage );
    MessagePopup ( "EGTSL", userMessage );
  }


  // unload the program
  if ( programId >= 0 )
  {
    ICT_Unload_Program (0, resourceId,
                           programId,
                           &errorOccurred, &errorCode, errorMessage);
  }
  // release ICT library
  if ( resourceId >= 0 )
  {
    ICT_Cleanup (0, resourceId, &errorOccurred, &errorCode, errorMessage);
  }


  // close resource manager
  RESMGR_Cleanup (0, &errorOccurred, &errorCode, errorMessage);



  return 0;
}
```

13.2 Starting Using the R&S EGTSL Loader
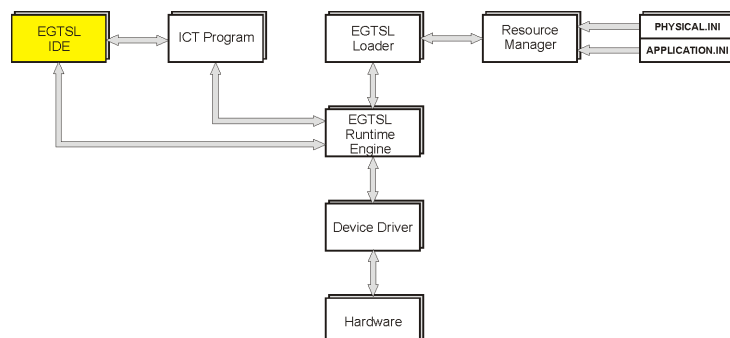
Figure 13-2: Opening R&S EGTSL IDE using R&S EGTSL Loader (software structure)

The R&S EGTSL Loader replaces the function calls from the ICT test library. In the R&S EGTSL Loader the names of the configuration files (`PHYSICAL.INI` and `APPLICATION.INI`) must be given. Hardware management is also performed using the resource manager. The R&S EGTSL runtime engine now opens the R&S EGTSL) and the corresponding ICT program. The ICT program opened can now be debugged in the R&S EGTSL user interface (R&S EGTSL). Using the R&S EGTSL runtime engine the ICT program can be executed on the test hardware directly from the R&S EGTSL user interface.

13.2.1 R&S EGTSL Loader, starting

To open the R&S EGTSL user interface (R&S EGTSL IDE) in the standalone mode, perform the following steps:

1.

EGTSL Loader

Start the R&S EGTSL Loader using **"Start -> Programs -> GTSL -> EGTSL"**
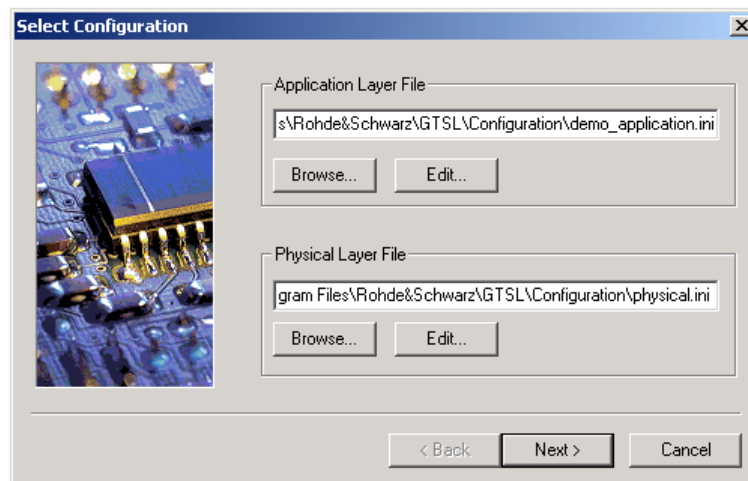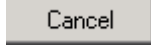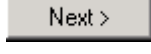
Figure 13-3: R&S EGTSL Loader Select Configuration

Note: If there is no valid license for the test libraries used, the R&S EGTSL user interface (R&S EGTSL IDE) is opened in the demo mode. The execution of the ICT programs opened is simulated.

2.   Using the dialog box (**"Browse..."** button), select the appropriate Application Layer Configuration File for the ICT program. The save path and the file name for the selected Application Layer Configuration File are displayed.

3.   Using the dialog box (**"Browse..."** button), select the appropriate Physical Layer Configuration File for the ICT program. The save path and the file name for the Physical Layer Configuration File selected are displayed.

Browse..	Opens the standard Windows dialog box for opening files. In the dialog box select the appropriate configuration file for the ICT program.
Edit...	The configuration file displayed is opened in the editor and can be edited.
Cancel	Quits the Loader.
Next >	Using the **"Next"** button, the given configuration is loaded. For details, see step 4

4. Using the **Next** button, load the configuration given.
   In accordance with the information from the configuration files given

   ● The user interface (IDE) is opened (continue with step 5) or
   ● The dialog box for the **vacuum control** is opened (see Figure 13-4).
     If, in the Application Layer Configuration File, there is an entry for the vacuum library in at least one bench, the dialog box for the vacuum control is opened
     Example:
     ```
     [bench->ict]
     ICTDevice1 = device->psam
     ICTDevice2 = device->pict
     ```
     VacuumControl1 = device->psys1
     VacuumControl2 = device->psys2
     ```
     SwitchDevice1 = device->pmb1
     ```
     ... etc.
     (continue with step 6)

5. In the R&S EGTSL user interface opened, it is now possible to load and debug and ICT program.

6. In the Vacuum Control window (see Figure 13-4) you can enter information on the vacuum control for the test adapter.
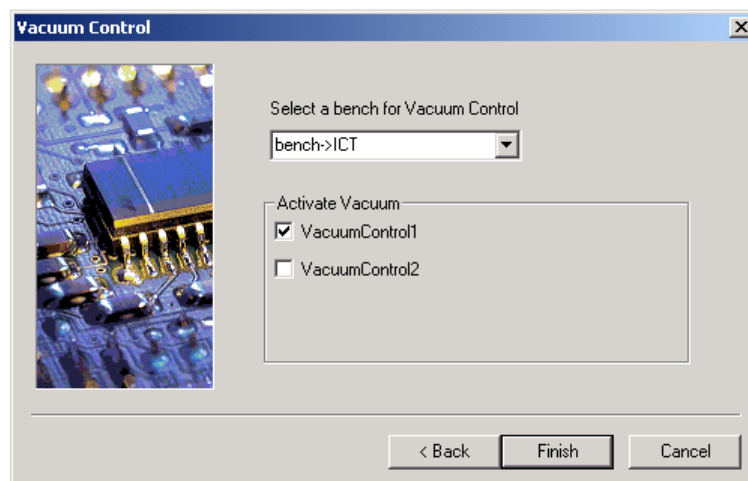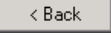
Figure 13-4: R&S EGTSL Loader Vacuum Control

7. Select the bench for the vacuum control.
   All benches for this Application Layer Configuration File with at least one vacuum control entry are displayed in the drop-down list box. In the list box select the bench that is to be relevant for the ICT program to be loaded.

8. Activate vacuum valves.
   The vacuum control entries in the selected bench are listed under **"Activate Vacuum"**. A maximum of four entries can be listed. By selecting the check box you can activate the corresponding vacuum valve.

9. ● [Finish]

   Using the **Finish** button, open the R&S EGTSL user interface (R&S EGTSL IDE).

   ● [< Back]

   The **Select Configuration** window (see Figure 13-3) is opened again.

   ● [Cancel]

   Quits the R&S EGTSL Loader.

Once the R&S EGTSL user interface has been quit, the opening dialog box (*R&S EGTSL Loader Select Configuration or R&S EGTSL Loader Vacuum Control*) is displayed again. Using the **"Cancel"** button the R&S EGTSL Loader is quit.

14 Report Format

For each test step, a report entry is created in the Report sub-window. This consists of the following entries:

- **Header**
  The following entries are listed in the header:
  - Test Step name
  - ID number
  - Part ID
    (the entry is only displayed if a Part ID is entered in the "Step Properties".)
  - Pass/Fail result
- **Measured result**
  The measured result consists of the measured value and the units as well as (in square brackets) the lower and upper limit.
  During the Diode and Transistor test steps, two lines with measured results are listed, because two measurements are carried out during these tests.
- **Error text** (displayed only in case the test step result was "Failed")
  In case the test step result was "Failed", a line with the error text is listed. The Contact, Continuity, Discharge and Short test methods display information on the pins. All other test methods list the "Report Text" from the "Step Properties".

Example:

Contact	`** 'Contact'[70] failed <----<<<`
	`2 [0 ... 0 ]`
	`P22, P27`
	
	Measured result: The number of the pins which have no contact is displayed. The lower and upper limits are zero.
	
	Error text: The error text displays the names of the pins which have no contact. A maximum of 10 pin names are displayed.

Continuity	`** 'GND track'[77] failed <----<<<`
	`1 [0 ... 0 ]`
	`P12 P13 / P89 P90`
	
	Measured result: The number of interruptions is displayed. The lower and upper limits are zero.
	
	Error text: The error text displays the names of the pins of the Contact test. Interruptions are marked with a slash.

Diode	```** 'D1'[76] (123.456) failed <----<<<```
	
	```-0.769865 V [-0.8 V ... -0.6 V]```
	
	```-0.00103737 uA [0.2 uA ... 1 uA]```
	
	```D1 failed```
	
	Measured result 1: The measured forward bias voltage (knee voltage) and, in brackets the lower and upper limit, are displayed.
	
	Measured result 2: The measured reverse bias current and, in brackets the upper and lower limit, are displayed.
	
	Error text: The text which has been entered in the "Step Properties" in the "Report Text" field is displayed.

Discharge	```** 'Discharge'[18] failed <----<<<```
	
	```1 [0 ... 0]```
	
	```Timeout; DCH pin 'P13'```
	
	Measured result: The number of the pins which could not be discharged is displayed. The lower and upper limits are zero.
	
	Error text: The error text displays the pin which was being discharged when the timeout occurred.

Fet	```** 'FET4'[71] passed```
	
	```4.9725 V [4.5 V ... 5.5 V]```
	
	```0.0213 V [-0.5 V ... 0.5 V]```
	
	Measured result 1: The measured drain-source voltage of the FET while the FET is in switched off state, and in brackets the lower and upper limits, are displayed.
	
	Measured result 2: The measured drain-source voltage of the FET while the FET is in switched on state, and in brackets the lower and upper limits, are displayed.

Impedance	```** 'Zx'[15] passed```
	```0.924677 uF [0.9 µF ... 1.1 µF]```
	
	Measured result: The measured value of the impedance measurement, and in brackets the upper and lower limit, are displayed.

Resistor	```** 'Rx'[30] (1234.5678.90) passed```
	```100.073 Ohm [99.9 Ohm ... 100.1 Ohm]```
	
	Measured result: The measured value of the resistance measurement and, in brackets the upper and lower limit, are displayed.

Short	`** 'Short 10'[71] failed <----<<<`
	`5 [0 ... 0 ]`
	`P12 P13 P6 / P89 P90`
	
	Measured result: The number of the pins involved in the short circuit is displayed. The lower and upper limits are zero.
	
	Error text: The error text displays the names of the pins which have mutually short circuited. Short circuits are separated from one another by a slash.

Transistor	`** 'Transistor'[18] passed`
	`0.618934 V [0.6 V ... 0.8 V]`
	`0.760869 V [0.6 V ... 0.8 V]`
	
	Measured result 1: The measured forward bias voltage (knee voltage) of the base-emitter diode, and in brackets the lower and upper limit, are displayed.
	
	Measured result 2: The measured forward bias voltage (knee voltage) the base-collector diode, and in brackets the lower and upper limit, are displayed.

Transistor Beta	`** 'TR5'[97] passed`
	`0.722228 V [0.42 V ... 0.78 V]`
	`245.464 [100 ... 300 ]`
	
	Measured result 1: The measured forward bias voltage (knee voltage) of the base-emitter diode, and in brackets the lower and upper limit, are displayed.
	
	Measured result 2: The measured current gain (beta), and in brackets the lower and upper limit, are displayed.

Zener Diode	`** 'ZD2'[95] passed`
	`27.3080 V [24.3 V … 29.7 V]`
	
	Measured result: The measured Zener working voltage, and in brackets the lower and upper limit, are displayed.

User-defined test	`'Filter'[15] failed`
	`-4.32 dB [-3.1 dB ... -2.9 dB]`
	`Filter attenuation test failed`
	
	Measured result: The measured value of the user-defined measurement, and in brackets the upper and lower limit, are displayed. The unit is specific to the user-defined test method.
	
	Error text: The error text is specific to the user-defined test method.

15 ICT Correction Data

15.1 System Residual Compensation

Due to the system residuals, the test system itself produces measuring errors. Using the program *ICTCorrection* all impedance measurements on a "typical pin" can be corrected. The data determined by the program are saved in an INI file and used for the correction of the measured values during the impedance measurements. As rule the *ICT Correction* program must be started only once for each test system configured. Only when the configuration is changed is it necessary to run the program again, e.g.:

- On the replacement of a
  - R&S TS-PSAM Source and Measurement Module
  - R&S TS-PICT In-Circuit Test Module
- On the addition or removal of TS-PMB Matrix Module B.
- On the addition or removal of modules that access the analog bus.

15.1.1 Normal Operation

ICTCorrection	Start the program *ICTCorrection* using **"Start -> Programs -> R&S GTSL -> Tools -> ICT Correction"**
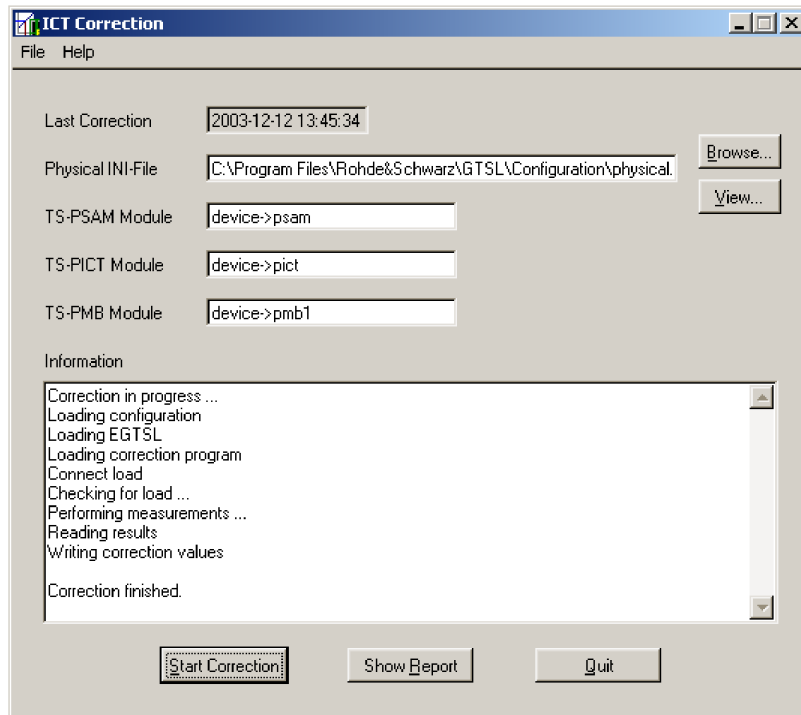
Figure 15-1: ICT Correction

(i) A special test adapter is required for the determination of the correction data.

The test adapter (ICTCorrection Load, SN. XXXX.XXXX.XX) is delivered together with the R&S TS-PICT In-Circuit Test Module.

The ICT-Correction must be executed only once configured with a TS-PMB module placed in a slot next to to TS-PICT module.

"Last Correction"	Indicates the date and time the last time the program ICT Correction was run.
"Physical INI-File"	Indicates the save path and the file name for the selected Physical Layer Configuration File for the test system to be calibrated. The information in the field can be edited.
Browse...	Opens the standard Windows dialog box for opening files. The Physical Layer Configuration File to suit the configured test system is selected in the dialog box.
View...	The Physical Layer Configuration File indicated is opened in an editor and can be edited.
"TS-PSAM Module"	The device entry (section) from the Physical Layer Configuration File for the R&S TS-PSAM Source and Measurement Module must be entered in this field. As a rule there is only one R&S TS-PSAM in the test system. If there are several R&S TS-PSAM modules, the module used for the determination of the correction data must be given.

"TS-PICT Module"	The device entry (section) from the Physical Layer Configuration File for the R&S TS-PICT In-Circuit Test Module must be entered in this field. As a rule there is only one R&S TS-PICT in the test system. If there are several R&S TS-PICT modules, the module used for the determination of the correction data must be given.
"TS-PMB Module"	The device entry (section) from the Physical Layer Configuration File for the R&S TS-PMB Matrix Module B must be entered in this field. As there are generally several R&S TS-PMB modules in the test system, the module for which the correction data are to be determined must be given.
"Information"	The program progress is indicated in this window. Any errors that occur are also displayed.
Start Correction	Starts the *ICTCorrection* program.
Show Report	Opens the report generated after the *ICTCorrection* program is run.
Quit	Quits the *ICTCorrection* program.

The following steps are necessary to determine the correction data:

1. Start *ICTCorrection* program.

2. Select the Physical Layer Configuration File for the test system to be calibrated.

3. Make device entries for the R&S TS-PSAM, R&S TS-PICT and R&S TS-PMB modules.

4. Using the **"Start Correction"** button, start the program.

5. When the program is running, you will be prompted to fit the special test adapter to the R&S TS-PMB module for which the correction data are to be determined.

6. If the program is run without errors, the file `ICTCorrection.ini` with the correction data is generated and saved.

7. Quit the program using the **"Quit"** button.

While the program is running, an Application Layer Configuration File is generated from the data given (Physical Layer Configuration File and device entries). Using the information from the Physical Layer Configuration File and the Application Layer Configuration File, a saved ICT program (`ICTCorrection.ict`) is opened automatically. Using this ICT program the correct function of the test adapter is checked and the correction data determined for various frequencies.

The following information is saved in the file ICTCorrection.ini :

● The values measured by the ICT program (`ICTCorrection.ict`)

● The correction factors determined

● The entries made in the *ICTCorrection* program

Before a new `ICTCorrection.ini` file is created, a backup of the previous version is saved as `ICTCorrection.bak`.

15.1.2 Silent Operation

The program `ICTCorrection.exe` version 1.13 or higher accepts two command-line parameters.

15.1.2.1 Parameter

Table 15-1: Silent Operation parameters

Parameter	Description
-silent	runs the program in silent mode without user inter-action
-proceed	the program calculates correction values even when measurements fail in silent mode

Syntax:

```
ICTCorrection [-silent][-proceed]
```

In silent mode, no user interaction is possible. Additionally the program will not show the user interface. Fit the test adapter "ICT Correction Load" to the configured switch module before starting the application in silent mode. When running the application in silent mode, the ICTCorrection.ini file must contain the following information:

```
[Report]
PhysicalIni = "C:\Rohde-Schwarz\GTSL\Configuration\physical.ini"
PSAM = "device->PSAM"
PICT = "device->PICT"
PMB = "device->PMB_6"
```

When executing the program with user interface, it generates this section automatically. In the main panel, you can enter the information.

15.1.2.2 Exit Codes

ICTCorrection returns one of the following exit codes in case of an error. Otherwise, it returns zero.

Table 15-2: Exit codes

Exit Code	Description
1	Init CVIRTE reports out of memory error
2	Init ActiveX failed
3	Could not allocate memory for ini file
4	Unable to load the panel
5	Key ICTCorrection not found in registry

Exit Code	Description
6	Unable to generate report file name
7	Device entry must have format 'device->...'
8	Error creating application layer ini file
9	Resource manager setup failed
10	Resource manager read ROM failed
11	Verify license failed
12	ICT setup failed
13	ICT load program failed
14	Error finding frame and slot of switch module
15	ICT run program failed
16	ICT write report failed
17	ICT Correction Load check failed
18	ICT measurements failed
19	ICT load detailed report failed
20	Invalid number of measurement results
21	Invalid result string
22	Unable to generate backup file name
23	Unable to backup correction ini file
24	Unable to read correction ini file
25	Unable to write correction ini file
26	Unrecognized parameter in command line

15.1.2.3 Programming Example

The following example shows how to execute ICTCorrection from a program written in C programming language.

```c
// Test program for executing ICTCorrection.exe in silent mode.
//
// Working directory must be: "C:\Program Files (x86)\Rohde-Schwarz\GTSL\EGTSL\Correction\"
//

#include <windows.h>
#include <ansi_c.h>

static char s_cmdLine[] =
  "C:\\Program Files (x86)\\Rohde-Schwarz\\GTSL\\EGTSL\\Correction\\ICTCorrection -silent";

static char s_currentDir[] =
  "C:\\Program Files (x86)\\Rohde-Schwarz\\GTSL\\EGTSL\\Correction";

int main()
{
  STARTUPINFO si;
  PROCESS_INFORMATION pi;

  ZeroMemory( &si, sizeof(si) );
  si.cb = sizeof(si);

  ZeroMemory( &pi, sizeof(pi) );

  // Start the child process.
  if( !CreateProcess( NULL,    // No module name (use command line)
      s_cmdLine,               // Command line
      NULL,                    // Process handle not inheritable
      NULL,                    // Thread handle not inheritable
      FALSE,                   // Set handle inheritance to FALSE
      0,                       // No creation flags
      NULL,                    // Use parent's environment block
      s_currentDir,            // Do not use parent's starting directory
      &si,                     // Pointer to STARTUPINFO structure
      &pi )                    // Pointer to PROCESS_INFORMATION structure
  )
  {
    printf( "CreateProcess failed (%d)\n", GetLastError() );
  }
  else
  {
    DWORD exitCode;

    // Wait until child process exits.
    WaitForSingleObject( pi.hProcess, INFINITE );

    // Get the exit code
    GetExitCodeProcess(pi.hProcess, &exitCode);

    printf("Exit Code %lu\n", exitCode);

    // Close process and thread handles.
    CloseHandle( pi.hProcess );
    CloseHandle( pi.hThread );
  }

  printf("Press <Enter> to terminate\n");
  getchar();

  return 0;
}
```

Figure 15-2: Programming Example

15.2 Adapter Compensation

Every adapter insert a capacitance into the measurement system, which need to be compensated. Using the 'ICT Fixture Compensation' allows to figure out these capacity values and to determine appropriate compensation values. Based on those compensation values and the initial determined correction values, the ICT program is updated to scope with those disturbances during the actual test runs. This procedure need to be done on every (empty) test-adapter, since every adapter has its own impedance as seen in the listed workflows.

The program "ICT Fixture Compensation" uses the "In-Circuit Test Library" (`ict.dll`) and the "Fixture Compensation Library (`fixtcomp.dll`) to perform the required steps. Of course also a test management software can use the DLLs to automate the

fixture compensation process. See also the R&S GTSL software description, chapter "Fixture Compensation Library".

In addition, following rules should be attended:

- If there are multiple adapters for one DUT, every adapter needs to be considered for its own. Therefore, multiple ICT Programs and Limits Files are generated, which results in a set of files for every adapter.

- For the Multiple Panel Testing, compensation values must be determined separately for every bench.

  ICT Fixture Compensation	Start the program *ICT Fixture Compensation* using **"Start -> Programs -> R&S GTSL -> ICT Fixture Compensation"**
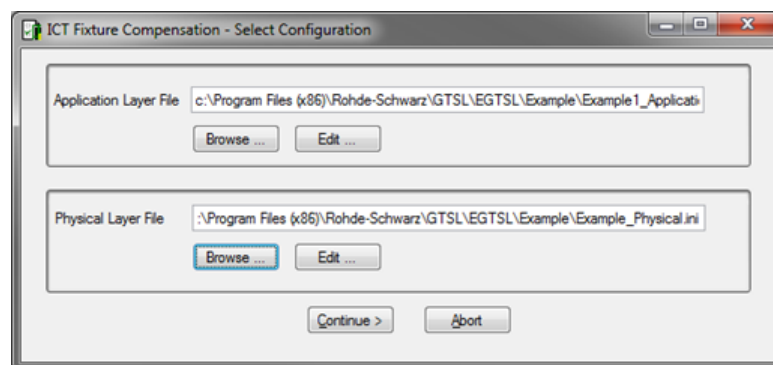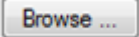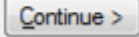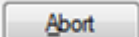
Figure 15-3: ICT Fixture Compensation Loader

Table 15-3: Application Layer File

"Application Layer File"	Indicates the loading path and the file name for an Application Layer Configuration File for the test system to be compensated. This information field can be edited.
"Physical Layer File"	Indicates the loading path and the file name for a Physical Layer Configuration File for the test system to be compensated. This information field can be edited.
	Opens the standard Windows dialog box for opening files. The current used folder is default-selected in the dialog box.
	Opens the standard Windows editor to edit/fix the current selected file.
	Checks the selected Application Configuration File as well as the Physical Configuration File and if both are valid, continues to the actual application.
	Closes the program immediately. Any change that is done by using the "Edit" Button on either of the configuration files will remain.
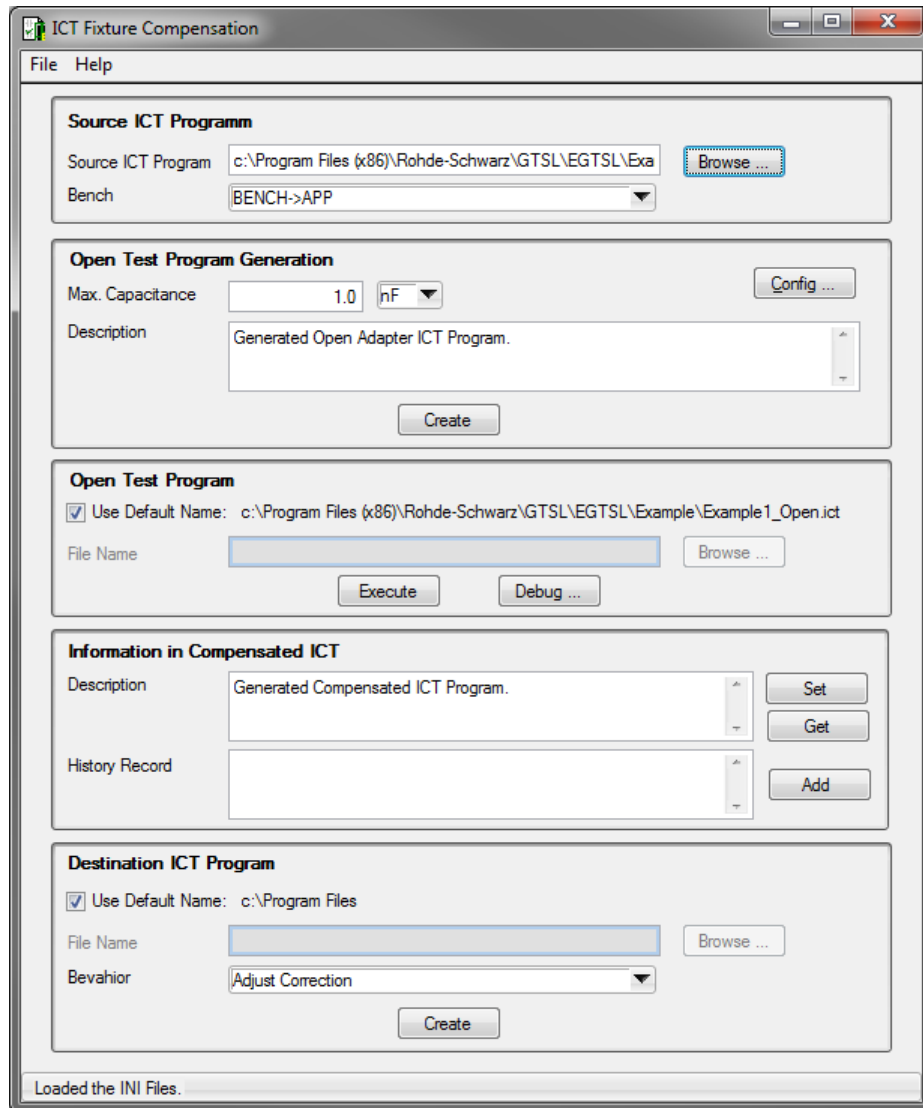
Figure 15-4: ICT Fixture Compensation Application

Table 15-4: Source ICT Program

"Source ICT Program"	Indicates the loading path and the file name for a valid and already debugged ICT program, which should be considered for the compensation measurement. This information field can be edited.
"Bench"	Lists all available benches from the Application Configuration File. This information is used for the execution later on and the default name generation.
Browse ...	Opens the standard Windows dialog box for opening files. The current used folder is default-selected in the dialog box.

Table 15-5: Open Test Program Generation
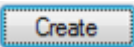
"Max Capacitance"	Indicates the maximum capacitance that is considered to compensate. All tests with nominal values greater than this value are ignored. This information field can be edited.
Config ...	Enables some advances options to be taken to customize the generated open adapter ict program. The dialog provides the possibility to enable/disable the short test as well as the discharge test, and to overwrite some limits. In fact the options are applied, when the Ok button is pressed.
"Description"	Indicates the description for the Open Adapter ICT (File that is used to measure the adapter faults). This information field can be edited.
Create	Creates a working ICT which can be used to measure with an open fixture adapter. This requires a valid description and max capacitance. The file is created with the default name.

Table 15-6: Open Test Program
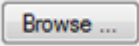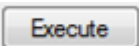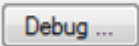
"Use Default Name"	Controls, if either the default file name for the open fixture measurement is taken into account, or a user-defined program is used instead. By unchecking (disabling the default mode) the "File Name" field becomes editable and the "Browse…" Button becomes enabled.
"File Name"	If no default name is used, this field indicates the loading path and the file name for a valid ICT Program that shall be used for the open adapter test. This information field can be edited.
Browse ...	If no default name is used, this control opens the standard Windows dialog box for opening files. The current used folder is default-selected in the dialog box.
Execute	Executes sub tests (Discharge, Short, In-Circuit) within the ICT. In succession, a report is written which is used later on to determine the correct compensation values. The filename for this report is derived from the Open ICT Program, where the extension is exchanged by ".txt". Every not passed Test will be ignored afterwards for the compensation process.
Debug ...	Debugs the sub tests (Discharge, Short, In-Circuit) within the ICT. **Since during debugging also invalid measurement can occur, the report generated by this step should not be used.** After successful debugging, execute the test with the "Execute" Button.

Table 15-7: Information in Compensated ICT
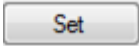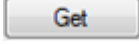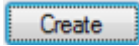
"Description"	Indicates the description for the Compensated ICT (File that is in the end produced by this tool). This information field can be edited.
Set	Set the text that is stated in the description field as description into the compensated ICT program. An empty description cannot be added, and must be therefore avoided.
Get	Receives the current set description in the Compensated ICT program. This is in first place the same as the description, which is established in the initial ICT.
"History Record"	Indicates a text that is used as a new history record in the compensated file. In contrast to the description property, can the user add an empty record, since then a default entry is produced. This information field can be edited.
Add	The information stated in the 'History Record' control will be added as new History Record. If the field is empty, when the process is started, a default entry stating the date and bench information is generated automatically.

Table 15-8: Destination ICT Program

"Use Default Name"	Controls, if either the default file name for the Compensated ICT Program is taken into account, or a user-defined program is used instead. By unchecking (disabling the default mode) the File Name field becomes editable and the "Browse…" Button becomes enabled.
"File Name"	If no default name is used, this field indicates the loading path and the file name for a valid ICT Program that shall be used. This information field can be edited.
"Behaviour"	For adding the compensation values, two different strategies are valid. On the one hand, ("Adjust Correction") the correction values can be taken into account and be adjusted if necessary. On the other hand, ("Not Adjust Correction") are compensation values were added directly.
Create	Adds compensation values by parsing the report file, which was generated after the open adapter measurement. In succession is the actual ICT program file written as filename, which is either the default name or a user specified name. In addition, a limits file is generated that holds information about the compensation and correction values.
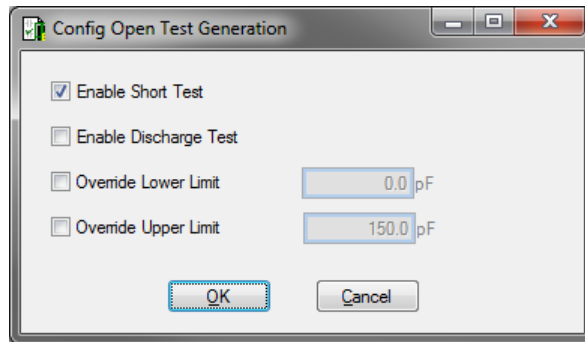
Figure 15-5: Open Test Generation configuration

Table 15-9: Open Test Generation configuration

"Enable Short Test"	If this option is set, the test "Short" of the incoming test is put into the intermediate ICT program for the 'Open Adapter Measurement'. (Default = on)
"EnableDischarge Test"	If this option is set, the test "Discharge" of the incoming test is put into the intermediate ICT program for the 'Open Adapter Measurement'. (Default = off)
"Override Lower Limit"	If this option is set, the default value for the lower limit is overwritten in the intermediate ICT program for the 'Open Adapter Measurement'. (Default = 0.0pF)
"Override Upper Limit"	If this option is set, the default value for the upper limit is overwritten in the intermediate ICT program for the 'Open Adapter Measurement'. (Default = 150.0pF)

A typical workflow to get a compensated ICT for one Adapter:

1. Start *ICT Adapter Compensation* tool.

2. Select the "Physical Layer Configuration File" as well as an "Application Layer Configuration File", according to the ICT program, which shall be compensated.

3. Click on the "Continue >" button to check the validity of the configuration files and open the actual application panel.

4. Select an already debugged ICT program choose the correct Bench, for which the compensation should be executed.

5. Fill in a maximum limit for the nominal test values and add a description for the open ICT program.

6. Create the Open Adapter ICT program file by clicking "Create".

7. Assure that the measurement adapter is empty.

8. Execute or debug the Open Adapter ICT Program.

9. Add additional information to the resulting compensated program. First add a new description; otherwise, the description of the initial ICT is used. The history record can be added on purpose.

10. Choose the correct behavior for the handling of the compensation and correction values.

11. Finally start the creation of the resulting compensated file by pushing the lower button "Create".

ⓘ During step 7 is an additional report file generated, which is necessary during addition of the compensation. This must be taken in consideration, if a user-defined Open ICT file is used, since the report file name must be equal to this file name except the extension.

A typical workflow to get compensated ICT's for multiple Adapters:

1. Generate a compensated file for the first adapter by following instruction in the workflow to gather compensation for one adapter.

2. For every additional adapter, follow those instructions again, whereby the compensated file of the first adapter should be used as base for the compensation process.

ⓘ The ICT Fixture Compensation application is able to handle already given Correction and Compensation values. This is due to the complexity of the correction value handling in the library, which automatically detects if any compensation value is established and use a different algorithm to integrate the correction value into the compensation calculation.

16 Error Messages

ⓘ Starting with GTSL 3.30, no GTSL license is required.

16.1 Compile Errors

Compile errors can occur in the following situations during the creation of the meta-code:

- At the opening of an ICT program.
- At the creation or running of test steps. During the execution of the **"Apply"** (or **"Auto apply"**) function, a compile error is displayed for an erroneous test step.

Compile errors are displayed as text in the Report sub-window. Erroneous test steps are marked in red in the Program sub-window.

The following list shows some common compile errors.

`Unknown pin X`	• A pin name was entered incorrectly. • The pin name is not listed in the `application.ini` file.
`<Parameter> is out of range`	The value of the parameter entered lies outside the permissible value range
`No complementary path found for pins X and Y`	For 6-wire measurements, the „even/odd rule" for Force and Sense pins was not adhered to. The entered pins have either both even or both odd pin numbers.
`Device type 'pict' is missing`	The test method requires a R&S TS-PICT module which was not configured in the bench as an ICT device.
`TS-LEGT license required`	For the test method, a R&S TS-LEGT license is required.

16.2 Runtime Errors

Runtime errors occur during the execution of the program and are displayed in separate error windows.

Examples:
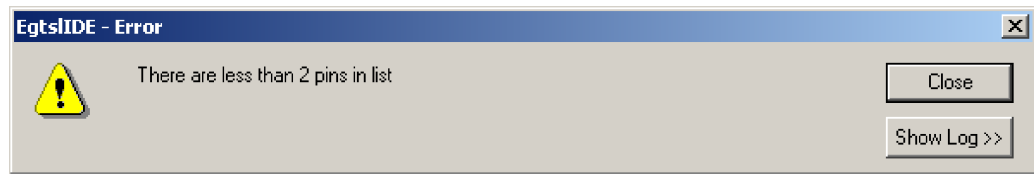
Figure 16-1: Runtime Error window (Example 1)

Show Log >>	With the **"Show Log"** button, an expanded view of the error message can be displayed.
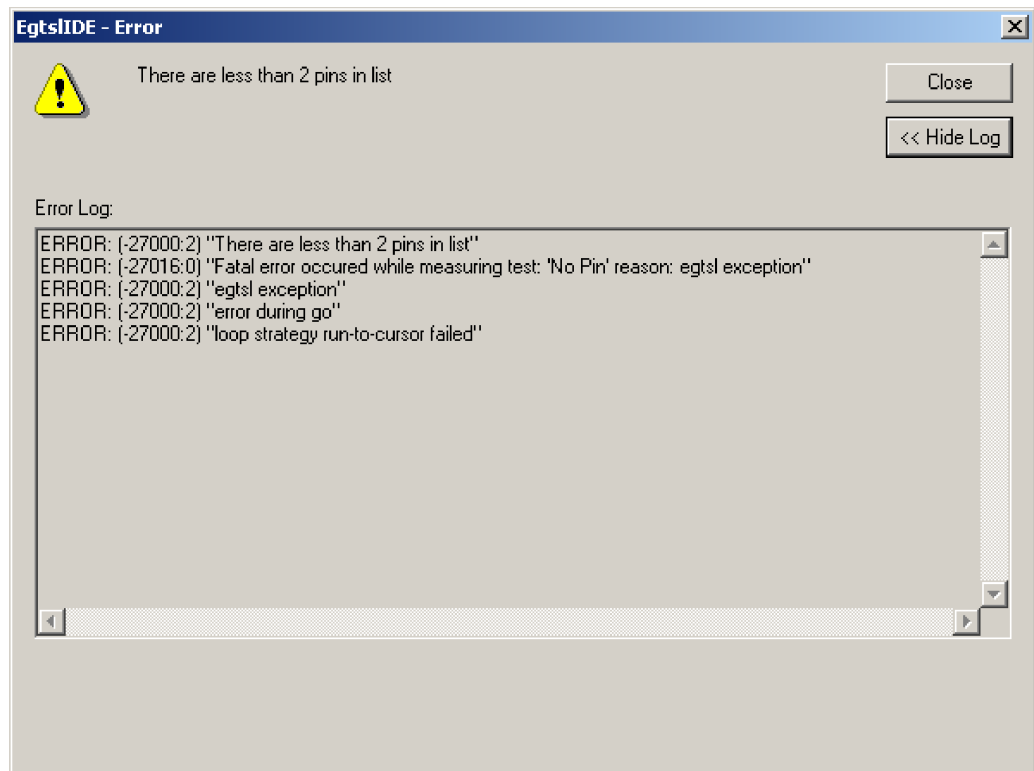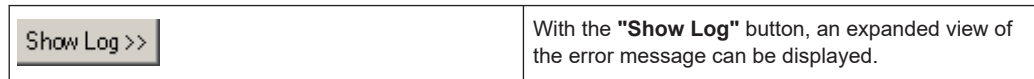
Figure 16-2: Error window, expanded view (Example 1)
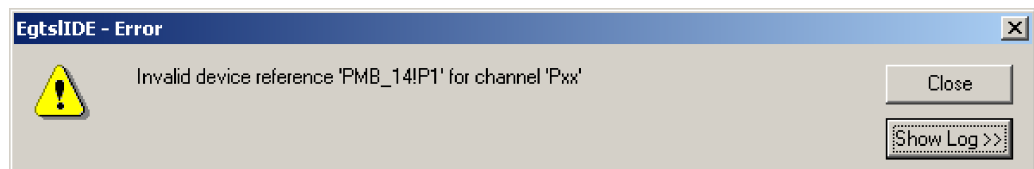
Figure 16-3: Runtime Error window (Example 2)

This difference between the two types of runtime errors is:

1. Runtime errors which frequently occur through incorrect entries and which can be easily corrected:

    ● `"There are less than 2 pins in list"`

The error occurs during the short circuit test or contact test if there are not at least two pins listed in the pin list.

- `"No pin in list"`
  The error occurs during the contact test or the discharge if the pin list is empty.

2. Error in `application.ini` and `physical.ini`

- `"Invalid device reference "PMB2!P1" for channel "XYZ""`
  The entered switch device PMB2 does not exist or is not configured as "Switch-Device..." for the bench.
- `"No channel id for channel "XYZ""`
  The physical channel name for "XYZ" does not exist (e.g. channel "P99" for a R&S TS-PMB)
- `"Invalid physical channel name "PMB2.P1" for channel "XYZ""`
  The syntax of the physical channel name (here: PMB2.P1) is incorrect. PMB2! P1 would be correct.

Other than the runtime errors shown above, most be hardware or software problems and should be reported to ROHDE & SCHWARZ.

17 ICT Extension Libraries

17.1 Overview

17.1.1 Objective

R&S EGTSL already offers a series of in-circuit test methods that can be used to test standard components such as resistors, capacitors, coils, semiconductors, etc. The ICT extension libraries also make it possible to develop own user-specific test methods and make them available in R&S EGTSL. Then special components can be tested within the R&S EGTSL in-circuit tests.

17.1.2 Functionality

User-specific test methods are fully integrated into R&S EGTSL so that they behave exactly like integrated test methods in terms of operation and functionality. All debugging options such as displaying the results of measurements in table format or graphically are also available for user-specific test methods. In addition, user-specific test methods make it possible to integrate additional measurement hardware into the in-circuit test. Of course the standard measurement modules of R&S EGTSL (R&S TS-PSAM, R&S TS-PICT and the matrix modules R&S TS-PMB) can also be used.

Parameters can be set for each user-specific test method covering a specifically defined number of properties. These properties could be setting parameters such as voltage, current or frequency. They could also be pin names for connecting measurement modules to the test object.

Sub-windows for user-specific test methods are shown in Chapter 5.5.13, "User-defined Test Methods", on page 107.

17.2 Using Extension Libraries

17.2.1 Configuration

Extension libraries must be configured before they can be used in an application. This is done by declaring them for the application in the Application Layer Configuration file (APPLICATION.INI).

The [extIct] section (see Chapter 7.1.6, "[ExtIct] Section", on page 140) contains one entry for each ICT extension library with the key word "UserDefinedDll", which contains the path and filename of the DLL.

Example:

```
[ExtIct]
UserDefinedDll = C:\IctExtensionDlls\MyOpAmp.dll
UserDefinedDll = %GTSLROOT%\EGTSL\ExtIct\RSSample\RSSample.dll
UserDefinedDll = MyRelay.dll
```

R&S EGTSL replaces the character sequence `%GTSLROOT%` with the installation path of R&S GTSL (normally `C:\Program Files\Rohde&Schwarz\GTSL`).

If only a filename is specified with no path, the DLL must reside in a directory that is included in the environment variable PATH.

When the R&S EGTSL user interface starts, the extension libraries are loaded and their corresponding icons appear in the toolbar.

17.2.2 Documentation

Every extension library has a documentation file associated with it, which must be in the same directory as the DLL itself. The documentation can be called from the R&S EGTSL user interface with the **Help** button in the **Info** sub-window of **Test Properties**.
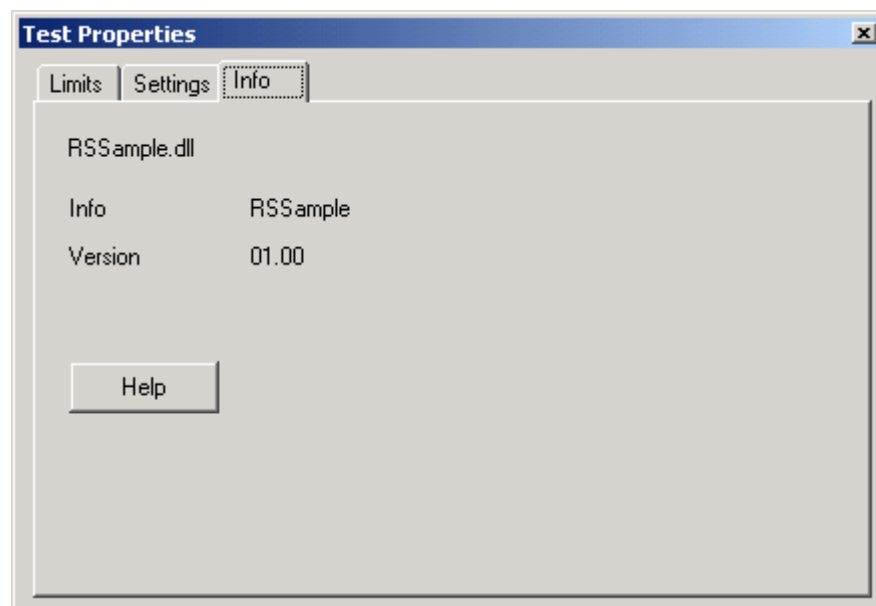
Figure 17-1: Test Properties Info

17.2.3 Inserting Test Steps

You can insert user-defined tests by clicking on the corresponding icon in the toolbar.

If the user-specific toolbar is not displayed, you can make it visible with the menu command <View><User-defined Toolbar>.

	The icon that is displayed is different for each user-specific test method. The same icon also appears in the program sub-window before the corresponding test step.
	If no separate icon is available for a test method, a standard icon is displayed.

It is also possible to insert user-defined tests with menu command
<Edit><Insert><Name of test method>.

17.2.4 Grouping Test Steps

To ensure the fastest possible execution time for the in-circuit test, user-defined test methods should be combined into groups of similar test methods if possible. The reason for this is that device settings are made when switching between different test methods, and this requires additional time (see also Chapter 17.3.7.5, "Prolog and Epilog", on page 290).

17.2.5 Running and Debugging Test Steps

Running and debugging test steps of user-defined test methods is no different than R&S EGTSL-internal test methods. The process described in Chapter 12, "Debugger", on page 225 also applies to this case.

17.3 Creating Extension Libraries

17.3.1 Overview

17.3.1.1 Software Structure
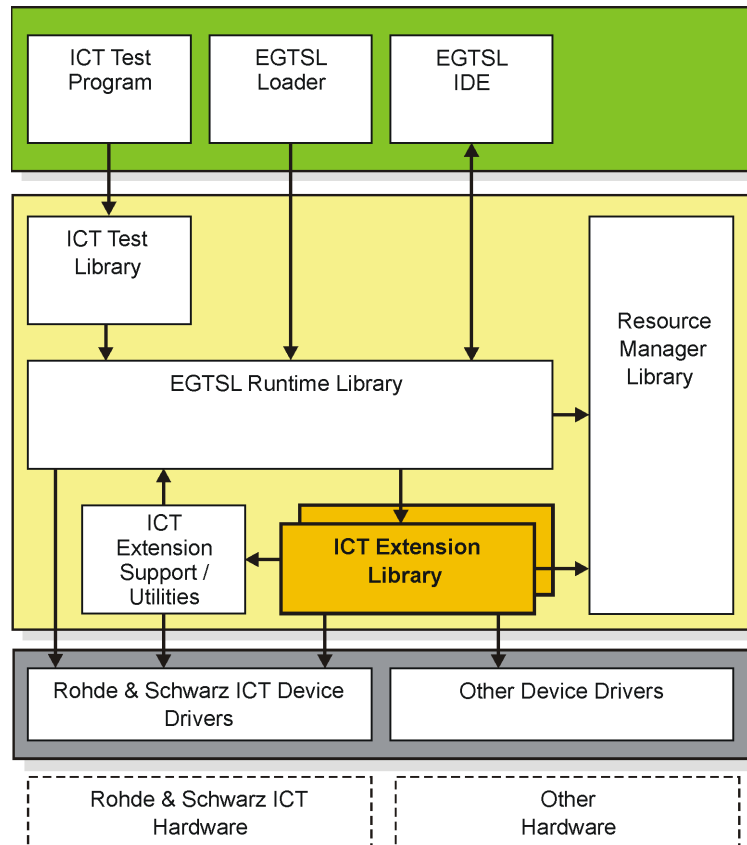
Figure 17-2: Software structure

The ICT extension libraries are loaded and called by the R&S EGTSL Run-time Library. The extension libraries use ICT hardware device drivers (R&S TS-PSAM, R&S TS-PICT, R&S TS-PMB) as well as other devices. The functions of the Resource Manager and ICT Extension Support Library can also be used.

17.3.1.2 Working Principle

Every ICT extension library is implemented as a Dynamic Link Library (DLL). This DLL makes a defined set of functions available that are called by the R&S EGTSL runtime library at the appropriate time. Of course the most important of these functions are responsible for performing the measurements and connections. The tasks of the

remaining functions are to initialize and prepare hardware modules, verify user entries and prepare the measurement value for display.

17.3.1.3 Integrating Hardware Modules

In the standard configuration, the ICT modules administered by R&S EGTSL, R&S TS-PSAM and R&S TS-PICT plus the matrix modules R&S TS-PMB are available for measurements and connections. They are already initialized and administered by R&S EGTSL, so there is no additional software overhead required in the ICT extension library.

The ICT Extension Support Library also offers functions for quickly connecting test object pins to the analog busses of the R&S CompactTSVP using the R&S TS-PMB matrix modules.

In addition, other measurement and stimulus modules can also be integrated. That could include R&S CompactTSVP modules such as the R&S TS-PFG function generator, but also any other modules or external devices such as power supply units.

17.3.1.4 File Structure

An ICT extension library generally consists of three files.

- The Dynamic Link Library (DLL), which makes functionality available.
- A bitmap file with the same name and filename extension .BMP, containing the icon associated with the test method.
- A file containing the documentation for the test method. Any file format can be selected (for example HTML, PDF, HLP, CHM), provided there is a program on the target system that is linked with that format and is able to display it.

When assigning names to ICT extension libraries, care must be taken to ensure the names are unique. We therefore recommend prefixing an abbreviation for your company to the beginning of the name. This is the reason the sample project included with delivery is called "RSSample", with RS standing for Rohde & Schwarz.

ICT extension libraries should be stored in a common directory (for example `C:\IctExtensionDlls\`). The directory should not be created within the R&S GTSL installation directory. This will prevent files from being overwritten when R&S GTSL is updated or reinstalled.

Since the full directory name is generally listed in the `APPLICATION.INI` file, make certain the directory is always in the same place for different test systems. Otherwise the ICT extension libraries cannot be loaded on another test system.

17.3.2 Sample Projects

Two sample projects for ICT extension libraries are stored in the source code in folder `…GTSL\EGTSL\ExtIct`.

17.3.2.1 RSSample

The RSSample project is a fully functional example of an ICT extension library using modules R&S TS-PSAM and R&S TS-PFG.

The R&S TS-PSAM module is an example of modules administered by R&S EGTSL, and is relatively easy to use. The R&S TS-PFG module is not administered by R&S EGTSL and therefore requires greater programming overhead in RSSample. The files for this project reside in folder …`GTSL\EGTSL\ExtIct\RSSample`.

File	Description
`RSSample.c`	Source code of RSSample library
`RSSample.dll`	Executable code of RSSample as Dynamic Link Library
`RSSample.lib`	Link library of RSSample
`RSSample.bmp`	Bitmap file for the library's icon
`RSSample.pdf`	Description of RSSample library for the user
`RSSample.prj`	LabWindows/CVI project RSSample
`RSSample.dsp`	Microsoft Visual Studio project RSSample
`ExtIctUtil.c, ExtIctUtil.h`	Source and header file for support functions (see Chapter 17.3.3.4, "Support Functions", on page 270)

17.3.2.2 RSSamplePSU

The RSSamplePSU project is similar to the RSSample project, but uses the R&S TS-PSU module instead of the R&S TS-PFG module.

This project is simpler than the RSSample project because the TS-PSU and TS-PSAM are both administered by R&S EGTSL.

The files for this project reside in folder `...GTSL\EGTSL\ExtIct\RSSamplePSU`.

17.3.2.3 Framework

The Framework project represent a framework for an ICT extension library which users can use to develop their own libraries.

Copy the files to a separate directory and change the filenames as appropriate to start the development of a new ICT extension library.

Places to be completed are identified by the comment

```
/* TO BE DONE ... */
```

File	Description
`Framework.c`	Source code of the frame
`Framework.bmp`	Bitmap file for the library's icon

File	Description
`Framework.prj`	LabWindows/CVI project Framework
`Framework.dsp`	Microsoft Visual Studio project Framework
`ExtIctUtil.c`, `ExtIctUtil.h`	Source and header file for support functions (see Chapter 17.3.3.4, "Support Functions", on page 270)

17.3.3 Internal Structure

Studying the two sample projects RSSample and Framework is a good way to learn the internal structure of an ICT extension library.

17.3.3.1 Project Structure

An ICT extension library can be created with different compilers. Project files for Lab-Windows/CVI and Microsoft Visual Studio for the two samples are included with delivery.

A LabWindows/CVI project for an ICT extension library contains the following files:

- Source file(s) for the extension library (`*.c`)
- Source file for support functions (`ExtIctUtil.c`)
- Header file for the extension library (`ExtIctTestDll.h`)
- Function panel for the support library (`ExtIctSupport.fp`)
- Function panel for the resource manager (`Resmgr.fp`)
- Function panel for each device driver that is used (`*.fp`)
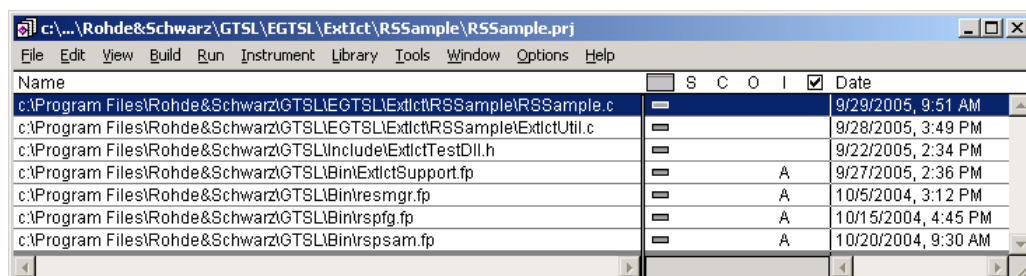
Figure 17-3: LabWindows CVI, project window

If the extension library is created with a compiler other than LabWindows/CVI, the Function Panel files (`*.fp`) must be replaced by the Link Libraries (`*.lib`) of the same name. The `*.fp` and `*.lib` files reside in folder `…\GTSL\Bin`.

The project must be configured so that a Dynamic Link Library (DLL) is generated.

With LabWindows/CVI the DLL export options are set so that the symbols specified in the include file will be exported:
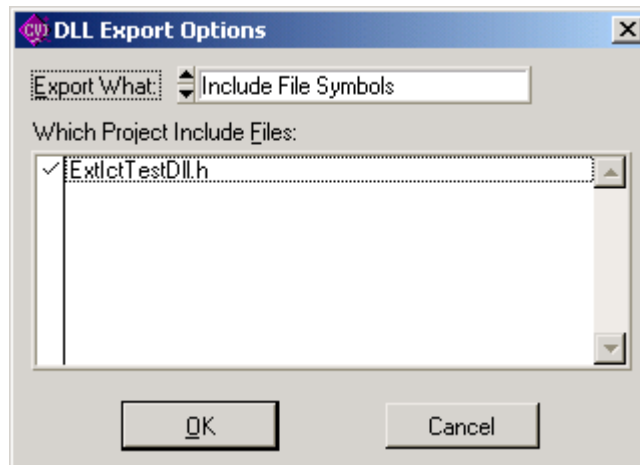
Figure 17-4: LabWindows/CVI, DLL Export Options

17.3.3.2 Export Interface

The export interface of an ICT extension library, i.e. the exported functions and variables, is identical for all extension libraries. These functions and variables are defined in the header file `ExtIctTestDll.h`. This file resides in folder `…\GTSL\include`.

The names of all exported functions and variables comply with the following naming convention:

"extict" + <subject> + <name>

"extict" is the name prefix for the ICT extension library, <subject> identifies the group to which the function belongs, for example "Info" for all variables or "Test" for some functions. <name> is the descriptive name of the function. The function performing the measurement is therefore given the name **extictTestMeasure( )**.

For functions that are listed to be correctly exported by the ICT extension library, the EXTICTTESTDLL_EXPORTS symbol must be defined before the include of the header file `ExtIctTestDll.h`:

```
#define EXTICTTESTDLL_EXPORTS
#include "ExtIctTestDll.h" /**< ExtIct  data types and export interface */
```

This definition ensures that the ICT extension library will **export** the functions and other code modules (which have not defined the symbol) will **import** the functions.

Each ICT extension library must implement all functions defined in `ExtIctTestDll.h`. Otherwise R&S EGTSL reports an error if one of the functions or variables is not present.

17.3.3.3 Data Structures

Special data structures are used to transfer parameters between R&S EGTSL and the ICT extension libraries. They are defined in `ExtIctTestDll.h`. The names of these data structures begin with the prefix "ExtIct", followed by the name of the data type.

Strings

There are three data structures that write strings of differing length:

ExtIctString	Strings with max. 256 characters, for example properties and values
ExtIctInfoString	Strings with max. 1024 characters, for error messages
ExtIctBigString	Strings with max. 4096 characters, for debug details.

```
typedef struct tagExtIctString
{
  char text[EXTICT_MAXSTRINGLENGTH];
} ExtIctString;
typedef struct tagExtIctInfoString
{
  char text[EXTICT_MAXINFOSTRINGLENGTH];
} ExtIctInfoString;
typedef struct tagExtIctBigString
{
  char text[EXTICT_MAXBIGSTRINGLENGTH];
} ExtIctBigString;
```

There are support functions for writing these strings to ensure the maximum length is not exceeded.

Status

The `ExtIctStatus` structure is composed of a numeric error code and an error message. This data structure is returned with most functions of the ICT extension library as the result.

An error code of = 0 means that a function has been successfully completed. Errors are reported as negative values.

In the case of success, the error message is empty. Otherwise the error text is entered.

```
typedef struct tagExtIctStatus
{
  int error;
  ExtIctInfoString message;
} ExtIctStatus;
```

Test Result

The structure `ExtIctTestResult` is composed of a status structure and the measurement result. This structure is returned as the result of function `extictTestMeasure`.

The structure indicates whether the measurement function was completed successfully. The error code is not set if a measurement value is outside the expected limits, however. Its purpose is rather to signal runtime errors.

If the measurement is successful, the variable "value" contains the result value. It is compared with the upper and lower limit by R&S EGTSL. The comparison result in an evaluation of whether a test step is a "Pass" or a "Fail".

In addition to the measurement value, the unit of measure (for example "mV", "kHz" or an empty string in some cases) is transferred in the variable "unit". The character sequence "info" can optionally be used to transfer non-numeric information. This information is added to the test report in the event a test fails (see also Section Chapter 17.3.8.2, "Non-numeric Test Results", on page 292).

```
typedef struct tagExtIctTestResult
{
  ExtIctStatus status;
  double value;
  ExtIctString unit;
  ExtIctString info;
} ExtIctTestResult;
```

Properties

The structure `ExtIctPropertyList` contains a list of all setting parameters and their values. This structure contains an array of elements of the structure `ExtIctPropertyItem`. The content of this structure is shown in the R&S EGTSL user interface (see Section 5.5.10.2).

`ExtIctPropertyItem` is a pair of two strings. The variable "name" contains the name of the property and "value" contains its value.

`ExtIctPropertyList` contains a list of 256 of these properties in addition to the nominal value ("nominal"), the unit ("unit") and the number of properties actually used ("size").

```
typedef struct tagExtIctPropertyItem
{
  ExtIctString name;
  ExtIctString value;
} ExtIctPropertyItem;
typedef struct tagExtIctPropertyList
{
  double nominal;
  ExtIctString unit;
  int size;
  ExtIctPropertyItem items[EXTICT_MAXNUMPROPERTIES];
} ExtIctPropertyList;
```

Administrative Data

When an ICT extension library is loaded and a test step is created, additional data structures are also created. Their main purpose is for storing internal data.

A structure `ExtIctTestInstance` is created for each test step. In addition to properties, it also contains a pointer to an internal data structure. This data structure is described in Section "SetupUserData" on page 273.

The structure `ExtIxtSetup` is created once for each test method. It contains the resource ID of the ICT program and also a pointer to an internal data structure. This data structure is described in Section "TestUserData" on page 273.

```
typedef struct tagExtIctTestInstance
{
  void *userData;
  ExtIctPropertyList properties;
} ExtIctTestInstance;
typedef struct tagExtIctSetup
{
  void *userData;
  int resourceId;
} ExtIctSetup;
```

17.3.3.4 Support Functions

Two support libraries are available to support ICT extension libraries:

- The ICT Extension Support Library (`ExtIctSupport.dll`)
- The ICT Extension Utilities (`ExtIctUtil.c`)

The **ICT Extension Support Library** offers functions of the R&S EGTSL Runtime Library for use in ICT extension libraries. This includes especially connections of matrix modules. The functions of this library are described in the corresponding online help `ExtIctSupport.hlp` in directory…`\GTSL\Bin`.

The **ICT Extension Utilities** offer functions to make it easy to handle special ExtIct data types. In some cases they also use the ICT Extension Support Library. By contrast, the ICT Extension Utilities are integrated into the source code. The functions of this library are described in source file `ExtIctUtil.c`. This file resides in folder …`\GTSL\EGTSL\ExtIct\Framework`.

To create a new ICT extension library, `ExtIctUtil.c` can be copied and easily expanded if additional functions are required.

`ExtIctUtil` contains functions for:

- Setting and chaining strings, taking into consideration the maximum length.
- Handling error messages.
- Converting property values (Strings) into numeric or Boolean values.
- Handling pin names.

17.3.3.5 Bitmap

Each ICT extension library can make its own icon available in a bitmap file. This icon is displayed in the R&S EGTSL user interface for each test step of this test method. The icon also appears in the toolbar.

The bitmap must be 16 x 16 pixels in size and can have a color depth of 16 or 256 colors. The lowest row of pixels is not displayed, i.e. the available surface is 16 x 15 pixels:
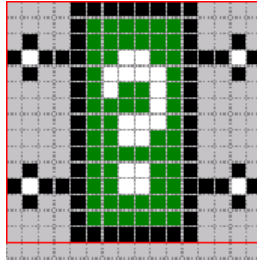
Figure 17-5: Bitmap Layout

If the bitmap file is not available, R&S EGTSL shows a default icon.

17.3.4   Description of Function

17.3.4.1   Header Files

An ICT extension library includes the following header files:

```
#define EXTICTTESTDLL_EXPORTS
#include "ExtIctTestDll.h" /**< ExtIct  data types and export interface */
#include "ExtIctSupport.h"     /**< ExtIct  support library */
#include "ExtIctUtil.h"        /**< ExtIct  utilities */


#include "resmgr.h"            /**< GTSL  Resource Manager */


#include "rspsam.h"            /**< TS-PSAM  device driver */
#include "rspfg.h"             /**< TS-PFG  device driver */
```

`ExtIctTestDll.h` exports the interface of the ICT extension library and defines the necessary data types.

`ExtIctSupport.h` and `ExtIctUtil.h` import the support functions.

`resmgr.h` imports the functions of the resource manager.

The corresponding header file is included for each device driver that is used, thus in the example above for R&S TS-PSAM and R&S TS-PFG.

The header files reside in folder …`\GTSL\include` with the exception of `ExtIctUtil.h`, which is in the project folder.

17.3.4.2   Exported Variables

Each ICT extension library exports the following variables, whose names begin with `extictInfo`:

```
/** test name */
EXTICTTESTDLL_API char exctictInfoTestName[] = "RSSample";

/** component version */
EXTICTTESTDLL_API char exctictInfoVersion[] = "01.00";

/** help file for this component */
EXTICTTESTDLL_API char exctictInfoHelpFileName[] = "RSSample.pdf";

/** component bitmap is loaded from a  file */
EXTICTTESTDLL_API int exctictInfoIconBitMap = -1;
```

exctictInfoTestName

This string variable contains the unique name of the test method. This name appears in the program sub-window with each test step and is displayed in the **Test Properties Info** window (see Figure 17-1).

To ensure the name is unique system-wide, we recommend making it begin with the company initials or abbreviation. This is the reason the sample project included with delivery is called "RSSample", with RS standing for Rohde & Schwarz.

exctictInfoVersion

This string variable contains the version number of the ICT extension library. It is displayed in the **Test Properties Info** window (see Figure 17-1).

exctictInfoHelpFileName

This string variable contains the name of the documentation file for the ICT extension library. This file must reside in the same directory as the DLL. You can display it by clicking the Help button in the **Test Properties Info** window (see Figure 17-1).

The documentation can be available in any file format. The only thing that is important is that a program that is called to display the file is linked with the file ending. Bringing up the documentation file from the R&S EGTSL user interface corresponds to double clicking this file in the Windows Explorer. The following common file formats are suitable for a documentation file:

File ending	Linked display program
.pdf	Adobe Reader
.html, .htm	Internet Explorer
.hlp	Windows Help
.chm	HTML Help Executable

exctictInfoIconBitMap

This integer variable normally contains the value -1. That value means that the icon for the ICT extension library is present as a *.BMP file. In that case R&S EGTSL loads the bitmap from the file.

If the ICT extension library is generated with Visual Studio, it is also possible to create the bitmap as a resource and add it to the DLL. In that case the resource ID of the bitmap resource is assigned to `exictInfoIconBitMap`. In that case R&S EGTSL extracts the bitmap from the DLL.

On the format of the bitmap, see Section Chapter 17.3.3.5, "Bitmap", on page 270.

17.3.4.3 Local Data Structures

An ICT extension library generally uses the following local data structures:

SetupUserData

This data structure is defined specifically for each ICT extension library. It contains information related to the entries in files `PHYSICAL.INI` and `APPLICATION.INI`. The data structure is dynamically allocated and filled in function `exictConfigSetup` and freed again in function `exictConfigCleanup`. A structure like this is created for each ICT program that is loaded.

In the RSSample project, the structure contains the following data items:

```
typedef struct
{
  int tracing;        /**< trace  flag */
  int simulation;     /**< simulation  flag */
  ViSession viPsam;   /**< session  handle for TS-PSAM (=0 in simulation mode) */
  ViSession viPfg;    /**< session  handle for TS-PFG (=0 in simulation mode)*/
  int lockedPfg;      /**< lock  flag for TS-PFG (=1 when locked) */
} SetupUserData;
```

`tracing` and `simulation` contain the values of the two entries "Tracing" and "Simulation" from file `APPLICATION.INI`.

`viPsam` and `viPfg`: For each hardware module that is used, the corresponding session handle of the device driver is stored in the structure. In Simulation mode, 0 is assigned to the handles.

`lockedPfg` is a flag that is created for each module not controlled by R&S EGTSL.

TestUserData

This data structure is defined specifically for each ICT extension library. It contains information related to the settings and measurement results of the test step. The main purpose of this structure is to move data between different processing steps of a test (compiling, measuring, displaying results).

The data structure is dynamically allocated in function `exictTestConstruct` and freed again in `exictTestDestruct`.

In the RSSample project, the structure contains the following data items:

```
typedef struct
{
```

```
/* Source property values */

E_SOURCE_MODE sourceMode;      /**< "Source  Mode" decoded property value */
double voltage;                /**< "Voltage"  property value, in Volts */
double offset;                 /**< "Offset"  property value, in Volts */
double frequency;              /**< "Frequency"  property value, in Hertz */

/* Measurement property values */
E_MEAS_FUNCTION measFunction;  /**< "Measurement  Function"
                                     decoded property value */
double range;                   /**< "Range"  property value in V, A, or Ohms */
int autoRange;                  /**< "Autorange"  property value - as Boolean */
double filter;                 /**< "Filter"  property value in Hertz */
double delay;                  /**< "Delay"  property value in seconds */
int average;                   /**< "Average"  property value */
double sampleInterval;         /**< "Sample  Interval" property value */

/* Measurement data */

E_MEAS_STATE measState;        /**< state  of the measurement */
double measuredValue;              /**< measured  value, dependent on measFunction,
                                   in units of nominal value */
double measuredRange;          /**< range  in which measurement was taken,
                                   in units of nominal value */
double scalingFactor;           /**< the  value by which the measured value must
                                     be multiplied to obtain the value in the
                                     desired unit */
double duration;               /**< duration  of measurement */
int simulationMode;            /**< 1  if simulation, 0 otherwise */

/* Switching information */

char busSourceHi [BUS_LEN];    /**< bus  name for Pin Source HI */
char busSourceLo [BUS_LEN];    /**< bus  name for Pin Source LO */
char busMeasHi   [BUS_LEN];    /**< bus  name for Pin Meas HI */
char busMeasLo   [BUS_LEN];    /**< bus  name for Pin Meas LO */

} TestUserData;  /**< test  user data */
```

The elements of this structure can mainly be divided into three groups:

Sections "Source property values" and "Measurement property values" contain values of the setting values already converted into target format. Function `exetictTestCompile` converts values and checks for valid value ranges. If the measurement function `exetictTestMeasure` is called, values stored here can be transferred directly to the device driver. This ensures optimal performance, since the conversion and check are eliminated each time the measurement is performed.

Section "Measurement data" contains data from the last measurement. The data is stored here in the function `exetictTestMeasure`. It is prepared for display in function `exetictTestGetDebugDetails`.

The last group, "Switching Information", contains information about the correct switching path. This information is also determined earlier during `exictTestCompile` so that it can be transferred directly to the device drivers for `exictTestMeasure`, thus guaranteeing a short execution time.

Table of Properties

All properties and their default values are combined together in a table. The table appears as follows in the RSSample project:

```
/** table with properties and default  values */
static const ExtIctPropertyItem s_properties[] =
{
  /* property name            default value */
  { "Source Mode",            "DC"        },
  { "Voltage",                "0.0"       },
  { "Offset",                 "0.0"       },
  { "Frequency",              "1000.0"    },
  { "Measurement Function",   "VDC"       },
  { "Range",                  "10.0"      },
  { "Autorange",              "On"        },
  { "Filter",                 "40.0e3"    },
  { "Delay",                  "0.0"       },
  { "Average",                "1"         },
  { "Sample Interval",        "1.0e-3"    },
  { "Pin Source HI",          ""          },
  { "Pin Source LO",          ""          },
  { "Pin Meas HI",            ""          },
  { "Pin Meas LO",            ""          },
};
/** number of property entries in s_properties  table */
#define NUM_PROPS (sizeof(s_properties)/sizeof(s_properties[0]))
```

On the left side are the names of properties in the order in which they are also displayed in the R&S EGTSL user interface. On the right side are their default values. When a new test step is created, all properties contain the corresponding default values. Because of that, the default values specified here must be valid values.

Values are generally stored as `ExtIctString`, i.e. as a string with a maximum of 256 characters. The data type for the individual properties is derived from the context, i.e. during the conversion and check in function `exictTestCompile`. Different types of data can be seen in the example, for example Voltage as a real number, Average as an integer, Autorange as a Boolean value and Source Mode as text.

Since it is not possible to specify default values for pin names, empty strings are assigned to them.

The `NUM_PROPS` macro automatically calculates the number of entries in the `s_properties` table.

A data type `PROP_INDEX` is defined for access to the property table. It defines constants for the indices of table entries. If changes are made to the `s_properties` table, `PROP_INDEX` must also be adjusted accordingly.

```
/** property enumeration = index  into s_properties
    must be kept consistent with the s_properties table  !
*/
typedef enum
{
  PROP_UNIT = -1,         /* special  value, see decodePropertyStringVal */

  PROP_SOURCE_MODE = 0,   /* 0 = first  index in s_properties table */
  PROP_VOLTAGE,
  PROP_OFFSET,
  PROP_FREQUENCY,
  PROP_MEAS_FUNCTION,
  PROP_RANGE,
  PROP_AUTO_RANGE,
  PROP_FILTER,
  PROP_DELAY,
  PROP_AVERAGE,
  PROP_SAMPLE_INTERVAL,
  PROP_PIN_SOURCE_HI,
  PROP_PIN_SOURCE_LO,
  PROP_PIN_MEAS_HI,
  PROP_PIN_MEAS_LO
} PROP_INDEX;
```

Tables for Value Conversions

As already explained in the previous section, values of properties are transferred as strings. The strings are not converted to their actual data type until during a test step is "compiled". The code module `extIctUtil.c` makes conversion functions available for numeric data types (int, double) that check the converted value against both an upper and a lower limit.

Properties that allow a selection of multiple strings as a value are treated with special conversion tables. Examples of these properties are mode settings ("Off", "AC", "DC") or physical units ("µV", "mV", "V"). A conversion table is created for each of these properties that assigns an integer number to each permissible string value.

```
/**
    @brief Property string value -> integer conversion  table.
    This table type holds pairs of permissible value strings
    and associated integer values. The table must end with an entry
    containing a NULL pointer for stringVal.
*/
typedef struct
{
  int        intVal;      /**< The  associated integer value */
  const char * stringVal;  /**< The  property value string */
} ExtIctUtilPropTableEntry;
```

In the RSSample project, for example, the following conversion tables are defined:

```
/** property value table for Source Mode */
static const ExtIctUtilPropTableEntry s_propsSourceMode[] =
{
  { SOURCE_MODE_AC,      "AC"  },
  { SOURCE_MODE_DC,      "DC"  },
  { SOURCE_MODE_OFF,     "OFF" },
  { 0,                   NULL  }  /**< always  terminate the table with a NULL entry */
};


/* property value table for Voltage units */
static const ExtIctUtilPropTableEntry s_propsVoltageUnits[] =
{
  /* scaling exponent,  unit */
  { 0,                   "V"   },
  { -3,                  "MV"  },
  { -6,                  "UV"  },
  { 0,                   NULL  }  /**< always  terminate the table with a NULL entry */
};
```

The last entry in the tables must contain a NULL string.

Table `s_propsSourceMode` assigns an "enum" constant to each mode. Table `s_propsVoltageUnits` assigns an exponent to each unit. The exponent is used to calculate the scaling factor. (1 mV = 10^{-3} V, 1 μV = 10^{-6} V).

Function `decodePropertyStringVal` from `extIctUtil.c` uses these tables to convert and check properties.

17.3.4.4 Configuration Functions

Configuration functions are used to prepare hardware for use or to take them to a defined state after use.

All configuration functions have a pointer to a data structure `ExtIctSetup` as a parameter and return `ExtIctStatus` as a result.

extictConfigSetup

```
ExtIctStatus extictConfigSetup ( ExtIctSetup * pSetup )
```

The Setup function is called by R&S EGTSL when a test program is loaded. R&S EGTSL transfers the resource ID of the test program to `pSetup->resourceId`.

Tasks performed by function `extictConfigSetup`

- Allocate and initialize data structure SetupUserData and save its pointer to pSetup.
- Determine configuration data from `APPLICATION.INI` and `PHYSICAL.INI` and store in SetupUserData, for example flags "Trace" and "Simulation".
- Determine session handles for hardware modules administered by and save in SetupUserData.
- Initialization of device drivers for hardware modules not administered by .

Very little new code needs to be created for these tasks, with the exception of the last. Only in the case of loading and initializing device drivers not administered by R&S EGTSL is more overhead required. See also Section Chapter 17.3.7, "Additional Hardware Modules", on page 287.

extictConfigCleanup

```
ExtIctStatus extictConfigCleanup ( ExtIctSetup * pSetup )
```

The Cleanup function is called by R&S EGTSL when a test program is unloaded.

Tasks performed by function `extictConfigCleanup`

- Closing device drivers for hardware modules not administered by R&S EGTSL.
- Freeing the data structure SetupUserData.

Handling of device drivers for hardware module not administered by R&S EGTSL is described in detail in Section Chapter 17.3.7, "Additional Hardware Modules", on page 287.

extictConfigProlog

```
ExtIctStatus extictConfigProlog ( ExtIctSetup * pSetup )
```

The purpose of the Prolog function is to prepare hardware modules to perform the next test step.

R&S EGTSL calls this function each time before executing a block of identical test steps.

Tasks performed by function `extictConfigProlog`:

- Call Prolog function of R&S EGTSL
- Locking of modules not administered by R&S EGTSL
- Reset of modules not administered by R&S EGTSL
- Preparing modules for the following test steps, for example closing the bus coupling relays, configuring the ground relay, switching the measurement and stimulus modules to the analog bus, any settings that are the same for all test steps.

See also Section Chapter 17.3.7.5, "Prolog and Epilog", on page 290.

extictConfigEpilog

The purpose of the Epilog function is to reset hardware modules to a defined state after the test steps have been performed.

R&S EGTSL calls this function each time after executing a block of identical test steps.

Tasks performed by function `extictConfigEpilog`:

- Reset of modules not administered by R&S EGTSL
- Unlock modules not administered by R&S EGTSL
- Call Epilog function of R&S EGTSL

See also Section Chapter 17.3.7.5, "Prolog and Epilog", on page 290.

exictConfigErrorCleanup

The purpose of the `ErrorCleanup` function is to reset hardware modules to a defined state after a runtime error.

R&S EGTSL calls this function when a runtime error has occurred in a test step of this ICT extension library.

Tasks performed by function `exictConfigErrorCleanup`:

Normally `exictConfigErrorCleanup` calls the two functions `extIctConfigEpilog` and `extIctConfigProlog` one after the other to make certain the hardware modules, coming from an unknown state, are prepared for the following test step.

17.3.4.5 Test-related Functions

The purpose of test-related functions is to check the settings for a test step, perform the test step and prepare the results for display.

exictTestConstruct

```
ExtIctStatus exictTestConstruct ( ExtIctTestInstance * pNewInstance,
                                  double * pLowerLimit,
                                  double * pUpperLimit  )
```

The purpose of the `Construct` function is to create data structures for a new test step.

R&S EGTSL calls this function if a new test step is inserted into the program or if a test step is loaded from an ICT file.

Tasks performed by function `exictTestConstruct`

- Allocate and initialize data structure `TestUserData` and save its pointer to `pNewInstance`.
- Copy content of table `s_properties` (property names and default values) to `pNewInstance`.
- Set default values for nominal value, unit, upper and lower limit.

exictTestDestruct

```
ExtIctStatus exictTestDestruct ( ExtIctTestInstance * pTestInstance  )
```

The purpose of the `Destruct` function is to free data structure `TestUserData`.

R&S EGTSL calls this function if a test step has been deleted from the program and when the ICT program is closed.

Tasks performed by function `exictTestDestruct`

- Free data structure `TestUserData`.

exictTestCompile

```
ExtIctStatus exictTestCompile ( ExtIctTestInstance * pTestInstance,
                                ExtIctSetup * pSetup )
```

The purpose of the `Compile` function is to "compile" a test step, i.e. check the properties for valid values and store the converted values in the `TestUserData` structure.

R&S EGTSL calls this function if a test step has been inserted or changed and when an ICT program is loaded.

Tasks performed by function `extIctTestCompile`

- Check whether the required hardware modules are present
- Check the values of properties to ensure they are valid
- Check the values of properties for consistency among each other
- Search for switching paths for matrix modules
- Prepare structure `TestUserData` to perform the test.

exictTestMeasure

```
ExtIctTestResult exictTestMeasure ( ExtIctTestInstance * pTestInstance,
                                    ExtIctSetup * pSetup )
```

The purpose of the `Measure` function is to configure and connect the instruments, perform the measurement and return the measurement results to R&S EGTSL.

R&S EGTSL calls this function when a test step is being performed.

Tasks performed by function `exictTestMeasure`

- Perform settings for hardware modules
- Create connection to test object
- Perform measurement
- Disconnect connection to test object
- Store data for "Debug Details" display in `TestUserData`
- Return measurement results to R&S EGTSL

exictTestGetDebugDetails

```
ExtIctBigString exictTestGetDebugDetails ( ExtIctTestInstance * pTestInstance  )
```

The purpose of the `GetDebugDetails` function is to prepare details of the last measurement for output in the Debug Details window.

R&S EGTSL calls this function when a test step is being displayed.

Tasks performed by function `exictGetDebugDetails`

- Prepare data from the last measurement from `TestUserData` for output.

The following data items can be collected during the measurement and sent to Debug Details:

- Status information from stimulus devices, for example current limiting active.

- Status information from measurement devices, for example overrange/underrange.
- The measurement range within which the measurement was performed (important for auto-ranging).
- Duration of the measurement.
- Any additional information that may help users to optimize and speed up measurements.

exictTestGetResultDescription

```
ExtIctString exictTestGetResultDescription ( ExtIctTestInstance * pTestInstance )
```

The purpose of the `GetResultDescription` function is to prepare the text for the "Nominal" column in the program sub-window.

R&S EGTSL calls this function when a test step is being displayed.
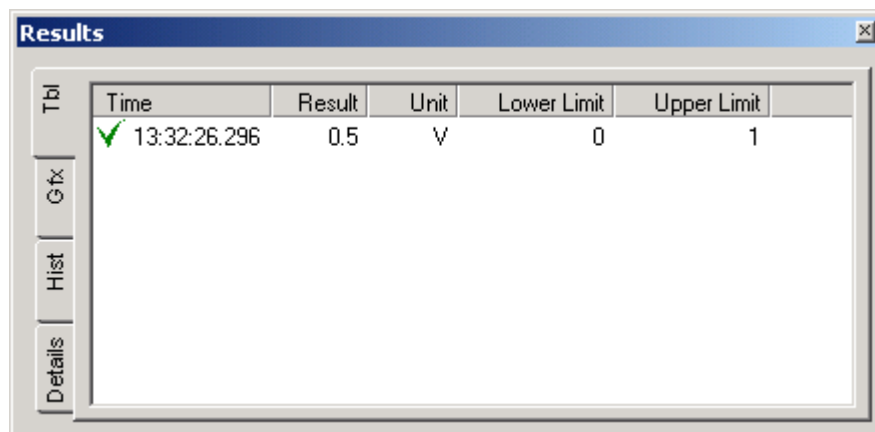
Tasks performed by function `exictGetResultDescription`

- Return nominal value and unit as a string

Special test cases that return a non-numeric result (for example the contact test) can return an empty string.

17.3.4.6 Functions for Displaying Results

These functions are responsible for generating the table in the "Result Tbl" sub-window.

Figure 17-6: Result Tbl sub-window

- Column heading ("Result") :
  `exictResultTableGetValueCaption`
- Result value ("0.5") :
  `exictResultTableFormatValueString`
- Unit ("V") :
  `exictResultTableFormatUnitString`

extictResultTableGetValueCaption

```
ExtIctString extictResultTableGetValueCaption ( void )
```

Function `extictResultTableGetValueCaption` returns the column heading of the table's second column in the "Result Tbl" sub-window.

- The column heading should indicate what a test method returns, for example "Frequency" or "Resistance".
- If a test method can return different results, for example RSSample, the default heading "Result" is used.

extictResultTableFormatValueString

```
ExtIctString extictResultTableFormatValueString ( double value )
```

Function `extictResultTableFormatValueString` returns the content of the table's second column in the "Result Tbl" sub-window as a string. The current measurement value is transferred in the parameter "value".

- Normally the measurement value is formatted and returned as text with a function such as `sprintf`.

extictResultTableFormatUnitString

```
xtIctString extictResultTableFormatUnitString ( char * unit  )
```

Function `extictResultTableFormatUnitString` returns the content of the table's third column in the "Result Tbl" sub-window as a string. The current unit is transferred in the parameter "value".

- Normally the unit that is transferred is returned as text.

17.3.5 Details of Implementation

17.3.5.1 Error Handling

Most functions of an ICT extension library return an `ExtIctStatus` structure as the result. The error code is set to 0 and the error message is set to an empty string at the beginning of each function.

```
ExtIctStatus status = { 0, "" };
```

If an error occurs while a function is running, function `setError` from `extIctUtil.c` can be used to transfer an error code and an error message:

```
setError ( &status, ERR_MEMORY_ALLOCATION_FAILED,
          "SetupUserData memory allocation failed" );
```

The constants for error codes are defined at the beginning of the C module. The beginning of the numbering is based on GTSL_ERROR_BASE_EXTICT_USER. Starting at this number, error codes are reserved specially for ICT extension libraries:

```
/* Error codes, based on GTSL_ERROR_BASE_EXTICT_USER = -300000 */
#define ERR_INTERNAL_ERROR          (GTSL_ERROR_BASE_EXTICT_USER  )
#define ERR_MEMORY_ALLOCATION_FAILED (GTSL_ERROR_BASE_EXTICT_USER  - 1)
#define ERR_MODULE_REQUIRED         (GTSL_ERROR_BASE_EXTICT_USER  - 2)
#define ERR_INVALID_NUM_PROPERTIES  (GTSL_ERROR_BASE_EXTICT_USER  - 3)
#define ERR_FUNCGEN_CONFIGURATION   (GTSL_ERROR_BASE_EXTICT_USER  - 4)
```

Error messages from R&S GTSL libraries such as the Resource Manager are forwarded directly to `setError`:

```
RESMGR_Lock_Device ( 0, pSetup->resourceId, RESMGR_KEY_FUNCTION_GENERATOR,
                     LOCK_TIMEOUT, &errorOccurred, &errorCode,  errorMessage );
if ( 0 != errorCode )
{
  setError ( status, errorCode, errorMessage );
}
```

Error messages from device drivers are somewhat more difficult to handle, since the driver only returns an error code. This error code must be converted into an error message in an additional driver call. A function of this type, `driverErrorXYZ` is implemented in the RSSample project for each device driver:

```
driverStatus = rspsam_ConfigureGround ( viPsam,  VI_TRUE );
if ( VI_SUCCESS != driverStatus )
{
  driverErrorPsam ( &status, viPsam, driverStatus, "rspsam_ConfigureGround"  );
}
```

To make certain the following code is no longer executed after an error, the code is divided into sections. The program checks whether an error has already occurred before each section is executed:

```
if ( 0 == status.error )
{
  RESMGR_Lock_Device ( 0, pSetup->resourceId, RESMGR_KEY_FUNCTION_GENERATOR,
                       LOCK_TIMEOUT, &errorOccurred, &errorCode,  errorMessage );
  if ( 0 != errorCode )
  {
    setError ( &status, errorCode, errorMessage );
  }
  else
  {
    deviceLocked = 1;
  }
}

if ( 0 == status.error )
{
  /* etc. */
}
```

Error messages are handled differently depending on the function that returns them (see also):

`exctictTestCompile`: The error message is generated as a "Compile Error" in the sub-window report.

`exctictTestMeasure`: The error is reported as a runtime error. Execution of the test is aborted. In this case the status of the hardware modules is undetermined. Because of this, `exctictConfigErrorCleanup` is called next.

`exctictConfigProlog`: The error is reported as a runtime error. Execution of the test is aborted.

`exctictConfigSetup`, `exctictTestConstruct`: The error is reported as a runtime error. The ICT program cannot be loaded.

An error message is ignored and not displayed in the remaining functions.

Error messages should preferably be generated in `exctictTestCompile`. They will be reported there as compile errors and that is where the user most expects to find them. Runtime errors should be avoided if possible, because they interrupt the test program that is running or may even prevent it from being able to load.

That means the values of all setting parameters should already be tested for the hardware's permissible upper and lower limits in `exctictTestCompile` to avoid a driver error while `exctictTestMeasure` is running.

We also recommend not reporting errors in `exctictConfigSetup` at all. Errors in `exctictConfigSetup` prevent the ICT program from being able to load. Then the user is no longer able to change the program to prevent the error from occurring. Errors in `exctictConfigSetup` in the RSSample project are only written to the trace file. If a hardware module cannot be configured or initialized, the only response is that its session handle is set to 0 in the `SetupUserData`. Then the `exctictTestCompile` function subsequently checks whether the session handles of the required device are non-zero. If they are equal to zero, a compile error is generated. This makes it possible to delay reporting errors that occur in `exctictConfigSetup` until later in `exctictTestCompile`.

17.3.5.2 Simulation

All R&S GTSL libraries as well as R&S EGTSL support a simulation mode. Users can turn simulation mode on in the `APPLICATION.INI` file. Simulation mode is also automatically set by R&S EGTSL, however, if there is no R&S TS-LEGT or R&S TS-LEG2 license in the system.

Simulation mode is a good option for demonstrating software if no hardware is available.

No device drivers may be called when simulation mode is active. Libraries simulate the hardware and return a simulated measurement result.

The ICT extension libraries should also implement simulation mode. For this purpose, the simulation flag is read in `exctictConfigSetup` and stored in the `ConfigUserData` structure:

```
pSetupUserData->simulation = EXTICTSUPPORT_SimulationMode  ( pSetup->resourceId );
```

The session handles of device drivers should also be set to 0 when simulation mode is active.

This will cause device drivers to be called in all subsequent `extict` functions only if no errors have occurred up to that point and the corresponding session handle is not 0:

```
if ( 0 == status.error && 0  != viPsam )
{
  driverStatus = rspsam_ConfigureGround ( viPsam,  VI_TRUE );
  if ( VI_SUCCESS != driverStatus )
  {
    driverErrorPsam ( &status, viPsam, driverStatus, "rspsam_ConfigureGround"  );
  }
}
```

In this manner, driver calls will not be executed in simulation mode.

Function `extictTestMeasure` should return the nominal value as a result in simulation mode. (The result of the measurement seems even more realistic if some "background noise" is overlaid).

```
if ( pSetupUserData->simulation )
{
  /* pass nominal value and unit as result in simulation mode */
  testResult.value = pTestInstance->properties.nominal;
  testResult.unit = pTestInstance->properties.unit;
  pTestUserData->measuredValue = testResult.value;
}
```

17.3.5.3 Tracing

In addition to simulation, tracing is a standard feature of R&S EGTSL. Trace outputs of all R&S GTSL and R&S EGTSL libraries are recorded in a central file.

ICT extension libraries should also support tracing, since it represents an easy method of troubleshooting. For this purpose, the trace flag is read in function `extictConfigSetup` and stored in the `ConfigUserData` structure:

```
int traceFlag = EXTICTSUPPORT_TracingEnabled (  pSetup->resourceId );

pSetupUserData->tracing = traceFlag;
```

If tracing is active, output can be generated with the `RESMGR_Trace` function.

```
if ( traceFlag )
{
  RESMGR_Trace ( ">>extictConfigSetup  RSSample" );
}
```

Formatted output is first formatted into a local buffer and then passed to RESMGR_Trace:

```
sprintf ( traceBuffer, "RSSample Version %s", extictInfoVersion  );
RESMGR_Trace ( traceBuffer );
```

For more information on trace output options, please refer to Online Help in the Resource Manager (file `RESMGR.HLP`).

17.3.5.4 Locking

All R&S GTSL libraries and R&S EGTSL are designed for multithreading. That means several threads can run in parallel within a process and hardware modules can be controlled by different threads. Since one hardware module cannot be used by two threads at the same time, however, there is an option to reserve it exclusively for a thread.

The Resource Manager makes the two functions `RESMGR_Lock_Device` and `RESMGR_Unlock_Device` available for this purpose. `RESMGR_Lock_Device` is used to request exclusive access to a hardware module. If the function returns successfully, no other thread is able to access the module, i.e. the same call from another thread will result in an error. `RESMGR_Unlock_Device` is used to unlock the module again.

ICT modules R&S TS-PSAM, R&S TS-PICT and matrix modules R&S TS-PMB in R&S EGTSL are locked as long as the in-circuit test is running. R&S EGTSL-internal Prolog and Epilog functions call `RESMGR_Lock_Device` and `RESMGR_Unlock_Device` for these modules.

ICT extension libraries should also implement these lock mechanisms for all hardware modules not administered by R&S EGTSL. To do this, `RESMGR_Lock_Device` must be called for each hardware module not administered by R&S EGTSL in `extictConfigProlog` and `RESMGR_Unlock_Device` must be called for each one in `extictConfigEpilog`.

17.3.6 R&S EGTSL-internal Hardware Modules

It is relatively easy to use hardware modules hardware modules administered by R&S EGTSL, R&S TS-PSAM, R&S TS-PICT and R&S TS-PMB, in an ICT extension library, since R&S EGTSL is already responsible for opening and closing the device drivers. The `ExtIctSupport` library makes functions available for determining session handles. The ICT extension library uses these session handles for all calls to the device driver.

The following additions to file `Framework.c` are all that is required to control an R&S EGTSL-internal hardware module:

- Add an `#include` for the device driver's header file
- Insert the device driver's `.FP` file into the project
- Call `EXTICTSUPPORT_GetSessionHandle` to determine the session handle
- Insert the device driver calls for settings and measurements in `extictTestMeasure`.
- If necessary, insert calls to device drivers for general settings in `extictConfigProlog`.

The R&S TS-PSAM module is used as an example of an R&S EGTSL-internal module in the `RSSample.c` project.

17.3.7 Additional Hardware Modules

Additional code must be created for hardware modules that are not administered by R&S EGTSL for example function generator R&S TS-PFG, since the device driver must be initialized and closed in the ICT extension library.

The R&S TS-PFG module is used as an example of a module monitored by R&S EGTSL in the `RSSample.c` project.

The following additions to file `Framework.c` are required to control a hardware module not administered by R&S EGTSL:

- Add an `#include` for the device driver's header file
- Insert the device driver's `.FP` file into the project
- Open and close the device driver in `extictConfigSetup` and `extictConfigCleanup`, taking into consideration the co-operative session concept (see Chapter 17.3.7.2, "Co-operative Session Concept", on page 288).
- Prepare the module settings in `extictConfigProlog` and reset the module in `extictConfigEpilog` (see Chapter 17.3.7.4, "Cleanup", on page 289).
- Insert the device driver calls for settings and measurements in `extictTestMeasure`.

17.3.7.1 Configuration of Hardware Modules

Before an ICT extension library can use a hardware module that is not administered by R&S EGTSL, it must first collect information about the module from the `APPLICATION.INI` file.

The hardware modules and switching modules administered by R&S EGTSL are declared with the keywords `ICTDevice` and `SwitchDevice`. Keywords must be defined separately for additional hardware modules. There are two ways to make this definition:

- For hardware modules that are also used in R&S GTSL libraries, keywords are available that are used by the libraries. Thus RSSample uses keyword **Function-Generator** for generator R&S TS-PFG because it is also used by R&S GTSL library FUNCGEN.
- New keywords can be defined specifically for hardware modules that are not used in R&S GTSL libraries. The name of the ICT extension library can be used as a prefix and a descriptive suffix can be added, for example **RSSampleSource.**

Example:

```
--- APPLICATION.INI ---

[bench->ICT]
Simulation              = 0
Trace                   = 1
ICTDevice1              = device->PSAM
ICTDevice2              = device->PICT
FunctionGenerator = device->PFG
```

```
; alternatively:
RSSampleSource    = device->PFG
SwitchDevice1        = device->PMB1
AppChannelTable      = io_channel->ICT



--- PHYSICAL.INI ---



[device->PFG]
Description       = "TS-PFG Module 1"
Type        = PFG
ResourceDesc = PXI7::12::INSTR
DriverDll         = rspfg.dll
DriverPrefix      = rspfg
DriverOption = "Simulate=0"
SFTDll            = sftmpfg.dll
SFTPrefix         = SFTMPFG
```

17.3.7.2 Co-operative Session Concept

Device drivers are generally opened in `extictConfigSetup` and closed in `extictConfigCleanup`. This applies only for the simple case, however, where they are not already open somewhere else in the test application. That case only occurs when a test program consists of both an in-circuit test and a function test part and the function test also uses the hardware module (for example the function generator R&S TS-PFG). Then the device driver cannot simply be opened and closed with no further consideration, since two instances of the same driver would then be open for the same hardware module.

The same rules thus apply to ICT extension libraries as to R&S GTSL libraries (see "Software Description R&S GTSL", Section 8.5.2). They must follow the co-operative session concept. That concept has two rules:

- Only one single instance of a driver may be open within an application for each hardware module.
- A device driver may only be closed if it is no longer required by any other R&S GTSL library or ICT extension library.

To ensure these rules are observed, opening and closing the device driver must be logged in to a central instance. The Resource Manager Library RESMGR is responsible for this task. It is used by all R&S GTSL and R&S EGTSL components.

Before a device driver can be opened, function `RESMGR_Open_Session` must be called. This function reports whether the device driver is already open. If it is, it returns the session handle for the device driver. If the driver hasn't been opened yet, it must be now. The session handle returned by the driver function must be transferred to the resource manager so it will be available for subsequent calls of `RESMGR_Open_Session`.

The process for closing a device driver is the same but in reverse order: First the driver session is logged out with `RESMGR_Close_Session`. This function reports whether any other sessions are still logged in. If any are, no further action is required. If the last session has just been logged out, however, the device driver must then be closed.

These actions are performed in the functions `extictConfigSetup` and `extictConfigCleanup`. In the `RSSample.c` project, sub-functions `openFuncGenSession` and `closeFuncGenSession` are responsible for these tasks.

17.3.7.3 Setup

Function `openFuncGenSession` in `RSSample.c` is called by `extictConfigSetup`. It uses function generator R&S TS-PFG as an example of how a hardware module not administered by R&S EGTSL must be initialized in an ICT extension library. An outline of the sequence follows:

1. Find the "FunctionGenerator" entry in the bench

2. Check if the device type is "PFG"

3. Lock the device

4. Open a resource manager session. If a session already exists, proceed with 9.

5. Get the resource descriptor (mandatory) from the configuration file

6. Get the option string (optional) from the configuration file

7. Initialize the device driver

8. Store the session handle in the resource manager

9. Unlock the device

17.3.7.4 Cleanup

Function `closeFuncGenSession` in `RSSample.c` is called by `extictConfigCleanup`. It uses function generator R&S TS-PFG as an example of how a hardware module not administered by R&S EGTSL must be closed in an ICT extension library. An outline of the sequence follows:

1. Lock the device

2. Close the resource manager session. If this was not the last session, proceed with step 4.

3. Close the device driver.

4. Unlock the device.

17.3.7.5 Prolog and Epilog

The purpose of the Prolog and Epilog functions is to place the hardware modules in a known state.

The functions are called by R&S EGTSL as soon as a block of identical test steps starts or ends.

Figure 17-7: ICT program with 3 blocks

A program is shown in Figure 17-7 that consists initially of three R&S EGTSL-integrated test steps (Contact, Short, Resistor). This is followed by three test steps of type "RSSample" and three test steps of type "MyOpAmp".

R&S EGTSL combines the test methods together in blocks to execute the program. All integrated test methods form a single block and each user-specific test method forms its own block.

Before a block is executed, its Prolog function runs, followed by the Epilog function as illustrated in Figure 17-7.
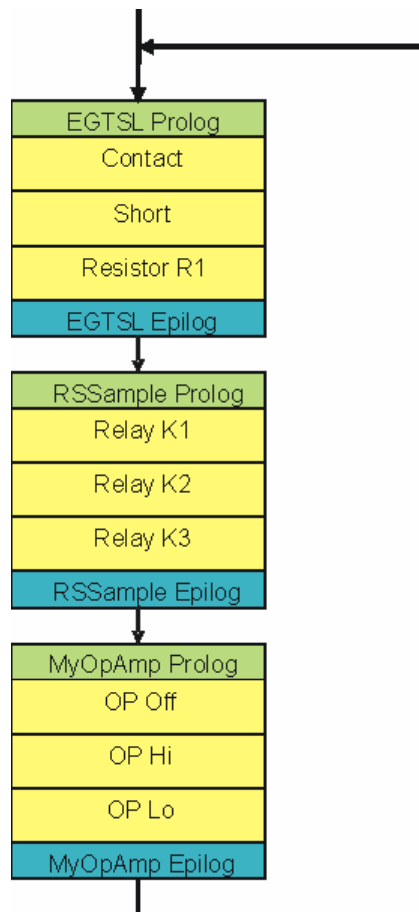
Figure 17-8: Block formation with Prolog/Epilog

Before the first test step (contact) is performed at the beginning of the in-circuit test, the status of the hardware modules is undetermined. The R&S EGTSL Prolog function must therefore run first to set the desired hardware state. Then the test steps of the first block (R&S EGTSL internal test methods Contact, Short, Resistor R1) can be performed. The R&S EGTSL Epilog function runs at the end of the first block. It again leaves the hardware modules in a defined state.

Before the block with the RSSample test steps, the Prolog function is called by RSSample. The hardware modules are reset at the end of the block using the RSSample Epilog function.

This is repeated for the block with the "MyOpAmp" test steps.

The Prolog and Epilog functions of a test method ensure in this manner that all necessary modules of a block are set correctly and that they assume a known state after the last test step.

To ensure an in-circuit test program runs as fast as possible, the test steps of ICT extension libraries must be combined together in blocks so that the Prolog/Epilog functions are not called more often than needed. In other words, blocks should be as long as possible.

17.3.8 Advanced Topics

17.3.8.1 Debugging ICT Extension Libraries

Because they are Dynamic Link Libraries rather than executable files, ICT Extension Libraries cannot be started directly in the development environment. To be able to debug an ICT extension library, the following settings must therefore be made:

For LabWindows/CVI

In the project under menu item **<Run><Select External Process…>**, enter file `…\GTSL\bin\EGTSLLoader.exe`.

For Visual Studio

In the project settings under **<Project><Settings><Debug>** enter file `…\GTSL\bin\EGTSLLoader.exe` as "Executable for debug session".

Then the R&S EGTSL Loader can be started from the development environment.

When making an "UserDefinedDll" entry in the `APPLICATION.INI` file, make certain the correct path is entered to the ICT extension library to be debugged.

17.3.8.2 Non-numeric Test Results

The measurement function of the ICT extension libraries returns a numeric value to R&S EGTSL as the result. It is also easy to imagine tests that do not return a numeric result. An example of this is the contact test in R&S EGTSL. In the event of an error, the contact test returns a list of pins that had no contact. In the case of "Pass", the list is empty.

In principle, however, the contact test also returns a numeric result like any other test that is compared against an upper and lower limit. A look at the report makes this clear:

```
** "Contact"[70] failed <----<<<
2 [0 ... 0 ]
P22, P27
```

The result of the contact test is a dimensionless number representing the number of pins without contact. The upper and lower limit value is 0, i.e. as soon as a pin without a contact is found, the test fails. In the case of "Fail", the contact test also returns a text containing the name of the pin that failed.

The same process can also be used in ICT extension libraries that return non-numeric results. The function `extictTestMeasure` returns the following results in this case:

- `testResult.value` returns the "number of errors"
- `testResult.unit` is an empty string
- `testResult.info` contains the test result as text in the case of "Fail"

The following adjustments are also required:

- 0 is assigned to the two limits in `exictTestConstruct`
- `exictTestGetResultDescription` returns an empty string, so the "Nominal" column in the display remains empty.

The text in `testResult.info` is only generated in the report by R&S EGTSL if the test step was "Fail". If `testResult.info` contains an empty string in this case, the text that is entered under "Step Properties, Test" is generated (see Figure 5-69).

17.3.8.3 Time Optimization

The goal of a good ICT extension library is to achieve the fastest measurement time possible. R&S EGTSL already offers the necessary support for this. The following items contribute to optimal performance:

- Separating the "compile" and "execution" of a test step
- Fast switching functions for R&S TS-PMB modules
- Block formation with Prolog/Epilog

The measurement should use as little time as possible for checking the setting values or converting strings into numeric values. Because of this, the "compile" and "measure" tasks are distributed over two different functions. The purpose of function `exictTestCompile` is to convert the user's entries and check them to make certain they are valid. Therefore it only needs to be called if a new test step is being entered or if an existing test step is being changed. If `exictTestCompile` has run with no errors, the values are correct and are available in the best format for further processing. Function `exictTestMeasure` does not require any additional tests. It forwards the checked values directly to the device driver.

The ICT Extension Support Library makes functions available to ensure rapid connection of R&S TS-PMB switching modules. Connection tasks are split up between `exictTestCompile` and `exictTestMeasure`. During the "compile" task, pin names are checked to ensure they are valid and the program searches for suitable switching path. The connections are made in `exictTestMeasure` by the ICT Extension Support Library without any time-consuming checks.

The distribution of tasks between the Prolog and the measurement function also contributes to the fastest possible execution time of measurements. Settings that are the same for all test steps of an ICT extension library can already be performed in the Prolog. Then they apply to all subsequent test steps. These settings also include above all connecting stimulus and measurement devices to the analogue bus. The purpose of the Epilog function is then to undo that connection. The more test steps are included in these blocks formed by Prolog/Epilog, the more time is saved.

17.3.8.4 Aspects of Compatibility

When developing an ICT extension library, there is no way to avoid making additions or correcting errors after ICT programs have already been written for the library. When this happens, a few points must be observed so programs that already exist can still be used.

If changes are made to an ICT extension library, the version number should be incremented in variable `extictInfoVersion`. The version number serves as information for the user, but it is not stored in the ICT program.

Existing property names must not be deleted or renamed. The names are stored in the ICT program and are assigned to the properties of an ICT extension library when the program is loaded. If it is not possible to make an assignment, an error message is generated and the ICT program cannot be loaded.

The order of entries in the property list is not critical. It can be changed without thereby making any existing ICT programs incompatible.

New properties can be added later. If an ICT program is loaded that does not contain these properties yet, its default values are assigned to it as defined in `extictTestConstruct`.

When an ICT program is saved, all of its properties are saved with it. This makes it impossible to load that program after that with an old version of the ICT extension library if a property has been added in the new version. ICT program are thus upwardly compatible, but not downwardly.

17.3.8.5 Communication between ICT Extension Libraries

Test steps in R&S EGTSL are created according to the concept so that they stand by themselves, i.e. independent of the preceding test steps of a measurement. If it should nevertheless become necessary to exchange information between the test steps of an ICT extension library or even between different libraries, the following options are available:

The most important functions of an ICT extension library have as a parameter a pointer to data structure `ExtIctSetup`. This data structure contains a pointer "userData" to an additional data structure that is the same for all test steps of an ICT as a extension library. Data elements to handle the exchange of information between test steps of the same ICT extension library can be created in this structure.

The shared memory of the resource manager library is suitable for data exchange between different ICT extension libraries. These functions can be used to create global data structures with system-wide unique names that can be exchanged between libraries.

Annex

A Example 1

ⓘ All files for example 1 are saved in the folder ...\GTSL\EGTSL\Example.

A.1 Circuit for Example 1
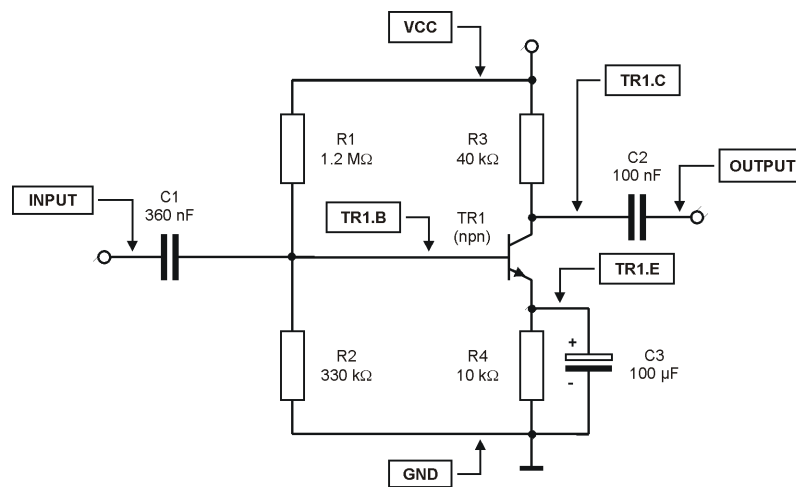
Figure A-1: Circuit, example 1

A.2 BDL File for Example 1

Example1.BDL

```
RESISTOR

NAME 'R1'
     VALUE 1.2 MOHM
     TOL+ 10%  TOL- 10%
     I_LIM 100.0 MA
     PIN_1 'VCC'
     PIN_2 'TR1.B'

NAME 'R2'
     VALUE 330 KOHM
     TOL+ 10%  TOL- 10%
     I_LIM 100.0 MA
```

```
                    PIN_1 'TR1.B'
                    PIN_2 'GND'


            NAME 'R3'
                    VALUE 40 KOHM
                    TOL+ 10%  TOL- 10%
                    I_LIM 100.0 MA
                    PIN_1 'VCC'
                    PIN_2 'TR1.C'


            NAME 'R4'
                    VALUE 10 KOHM
                    TOL+ 10%  TOL- 10%
                    I_LIM 100.0 MA
                    PIN_1 'TR1.E'
                    PIN_2 'GND'


        CAPACITOR


        NAME 'C1'
                    VALUE 360 NF
                    TOL+ 10%  TOL- 10%
                    PIN_1 'INPUT'
                    PIN_2 'TR1.B'


        NAME 'C2'
                    VALUE 100 NF
                    TOL+ 10%  TOL- 10%
                    PIN_1 'TR1.C'
                    PIN_2 'OUTPUT'


        POL_CAP


        NAME 'C3'
                    VALUE 100 UF
                    TOL+ 10%  TOL- 10%
                    PIN_A 'TR1.E'
                    PIN_C 'GND'


        TRANSISTOR


        NAME 'TR1'
                    NPN
                    UBE 0.6 V
                    TOL+ 30%  TOL- 30%
                    PIN_E 'TR1.E'
                    PIN_B 'TR1.B'
                    PIN_C 'TR1.C'


        NODE
```

```
NAME 'TR1.E'
     DEV_NAME 'R4' PIN_1
     DEV_NAME 'C3' PIN_A
     DEV_NAME 'TR1' PIN_E

NAME 'TR1.B'
     DEV_NAME 'R1' PIN_2
     DEV_NAME 'R2' PIN_1
     DEV_NAME 'C1' PIN_2
     DEV_NAME 'TR1' PIN_B

NAME 'TR1.C'
     DEV_NAME 'R3' PIN_2
     DEV_NAME 'C2' PIN_1
     DEV_NAME 'TR1' PIN_C

NAME 'GND'
     DEV_NAME 'R2' PIN_2
     DEV_NAME 'R4' PIN_2
     DEV_NAME 'C3' PIN_C

NAME 'OUTPUT'
     DEV_NAME 'C2' PIN_2

NAME 'INPUT'
     DEV_NAME 'C1' PIN_1

NAME 'VCC'
     DEV_NAME 'R1' PIN_1
     DEV_NAME 'R3' PIN_1
```

A.3 ICT Report Generated for Example 1

Example1_Report.txt

```
ATG REPORT
==========
Created: 2005-12-27 15:00:08
ATG Version: ATG 02.10
EGTSL Version: EGTSL 02.10
Input:
------
Board Description: 'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example1.BDL'
Physical Layer:    'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example_Physical.ini'
Output:
```

```
-------
ICT Program:      'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example1.ict'
Application Layer: 'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example1_Application.in
Report:           'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example1_Report.txt'
BDL Parser: 0 error(s) and 0 warning(s).
BDL cross check: OK.
>>> Message from TR1 <<<
TS-PSU not configured in 'physical.ini', simple transistor test inserted
Test coverage report
Number Weight Coverage
Components 8
Regular tests 7 1 87.5%
Reduced tests 1 0.5 6.25%
Short/Contact test only 0 0.1 0%
Total test coverage 93.8%
Wiring List info: Nodes mapped to a total number of 7 pin(s).
```

A.4 Application Layer Configuration File Generated for Example 1

Example1_Application.ini

```
[ResourceManager]
; general trace settings (normally off)
Trace          = 0
TraceFile      = %GTSLROOT%\resmgr_trace.txt


;------------------------------------------------------------------------------
[LogicalNames]
ICT   = bench->ICT


;------------------------------------------------------------------------------
[bench->ICT]
Description     = ICT bench (Simulation)
Simulation      = 1
Trace           = 0
ICTDevice1      = device->psam
ICTDevice2      = device->pict
SwitchDevice1   = device->pmb1
AppChannelTable = io_channel->ICT
AppWiringTable  = io_wiring->ICT


[io_channel->ICT]
GND           = pmb1!P1
INPUT         = pmb1!P2
OUTPUT        = pmb1!P3
TR1.B         = pmb1!P4
TR1.C         = pmb1!P5
```

```
TR1.E          = pmb1!P6
VCC            = pmb1!P7


;-------------------------------------------------------------------------------


[io_wiring->ICT]
GND            = F1 S15 X10A1
INPUT          = F1 S15 X10A2
OUTPUT         = F1 S15 X10A3
TR1.B          = F1 S15 X10A4
TR1.C          = F1 S15 X10A5
TR1.E          = F1 S15 X10A6
VCC            = F1 S15 X10A7


;-------------------------------------------------------------------------------
```

B Example 2

All files for example 2 are saved in the folder ...\GTSL\EGTSL\Example.

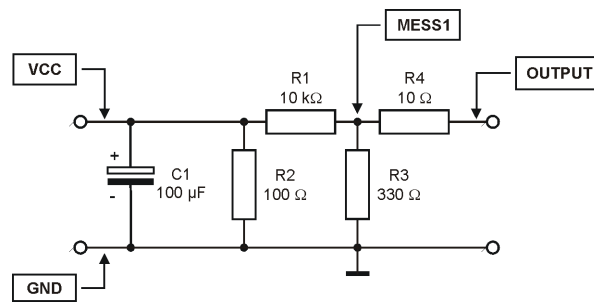## B.1 Circuit for Example 2

Figure B-1: Circuit, example 2

B.2 BDL File for Example 2

Example2.BDL

```
RESISTOR

NAME 'R1'
  VALUE 10 KOHM
  TOL+ 10% TOL- 10%
  I_LIM 100.0 MA
  PIN_1 'VCC'
  PIN_2 'MESS1'

NAME 'R2'
  VALUE 100 OHM
  TOL+ 10% TOL- 10%
  I_LIM 100.0 MA
  PIN_1 'VCC'
  PIN_2 'GND'

NAME 'R3'
  VALUE 330 OHM
  TOL+ 10% TOL- 10%
  I_LIM 100.0 MA
  PIN_1 'MESS1'
  PIN_2 'GND'
```

```
NAME 'R4'
  VALUE 10 OHM
  TOL+ 10% TOL- 10%
  I_LIM 100.0 MA
  PIN_1 'MESS1'
  PIN_2 'OUT'

POL_CAP

NAME 'C1'
  VALUE 100 UF
  TOL+ 10% TOL- 10%
  PIN_A 'VCC'
  PIN_C 'GND'

NODE

NAME 'GND'
  DEV_NAME 'C1' PIN_C
  DEV_NAME 'R2' PIN_2
  DEV_NAME 'R3' PIN_2

NAME 'OUT'
  DEV_NAME 'R4' PIN_2

NAME 'VCC'
  DEV_NAME 'C1' PIN_A
  DEV_NAME 'R1' PIN_1
  DEV_NAME 'R2' PIN_1

NAME 'MESS1'
  DEV_NAME 'R1' PIN_2
  DEV_NAME 'R4' PIN_1
  DEV_NAME 'R3' PIN_1
```

B.3   ICT Report Generated for Example 2

Example2_Report.txt

```
ATG REPORT
==========

   Created:        2005-12-27 15:02:19
   ATG Version:    ATG 02.10
   EGTSL Version:  EGTSL 02.10
```

```
Input:
------

Board Description:  'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example2.BDL'

Physical Layer:     'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example_Physical.ini'


Output:
-------

ICT Program:        'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example2.ict'

Application Layer:  'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example2_Application.i

Report:             'C:\Program Files\Rohde&Schwarz\GTSL\EGTSL\Example\Example2_Report.txt'




BDL Parser: 0 error(s) and 0 warning(s).


BDL cross check: OK.


>>> Message from R1 6-Wire Test Proposal          <<<
Total error for 3-wire mode: 23.351%
              6-wire mode: 3.1255%


>>> Message from R1                               <<<
Test only possible by exceeding the error limits
Guarding error limiting test quality


>>> Message from R4 4-Wire Test Proposal          <<<
Total error for 2-wire mode: 21.576%
              4-wire mode: 1     %


>>> Message from R4                               <<<
Test only possible by exceeding the error limits
Test limits include system residuals of 2.124 Ohms


Test coverage report
                      Number  Weight  Coverage
Components               5
Regular tests            5      1      100%
Reduced tests            0     0.5      0%
Short/Contact test only  0     0.1      0%
Total test coverage                    100%


Wiring List info: Nodes mapped to a total number of 8 pin(s).
```

B.4 Application Layer Configuration File Generated for Example 2

Example2_Application.ini

```
[ResourceManager]
; general trace settings (normally off)
Trace          = 0
TraceFile      = %GTSLROOT%\resmgr_trace.txt


;-------------------------------------------------------------------------------
[LogicalNames]
ICT   = bench->ICT


;-------------------------------------------------------------------------------
[bench->ICT]
Description     = ICT bench (Simulation)
Simulation      = 1
Trace           = 0
ICTDevice1      = device->psam
ICTDevice2      = device->pict
SwitchDevice1   = device->pmb1
AppChannelTable = io_channel->ICT
AppWiringTable  = io_wiring->ICT


[io_channel->ICT]
GND            = pmb1!P1
MESS1          = pmb1!P2
OUT            = pmb1!P3
VCC            = pmb1!P4


; nodes used for test proposals, which may be deleted, if the proposals are discarded
; ------------------------------------------------------------------------------
GND.Sense        = pmb1!P6
MESS1.Sense      = pmb1!P5
OUT.Sense        = pmb1!P7
VCC.Sense        = pmb1!P9


;-------------------------------------------------------------------------------

[io_wiring->ICT]
GND            = F1 S15 X10A1
MESS1          = F1 S15 X10A2
OUT            = F1 S15 X10A3
VCC            = F1 S15 X10A4


; nodes used for test proposals, which may be deleted, if the proposals are discarded
; ------------------------------------------------------------------------------
GND.Sense        = F1 S15 X10A6
MESS1.Sense      = F1 S15 X10A5
OUT.Sense        = F1 S15 X10A7
VCC.Sense        = F1 S15 X10A9


;-------------------------------------------------------------------------------
```