R&S®GTSL
Debug Driver Server Software
User Manual

1179136402
Version 02

ROHDE&SCHWARZ
Make ideas real

This document is valid for the following software versions.

- R&S®GTSL version 3.42 and higher

Muehldorfstr. 15, 81671 Muenchen, Germany
Phone: +49 89 41 29 - 0
Email: info@rohde-schwarz.com
Internet: www.rohde-schwarz.com

1179.1364.02 | Version 02 | R&S®GTSL

Throughout this manual, products from Rohde & Schwarz are indicated without the ® symbol , e.g. R&S®GTSL is indicated as R&S GTSL.

Contents

1 Documentation Overview.....5
2 Software Installation.....6
3 System Overview.....7
4 Quick-Start.....9
5 R&S GTSL Instrument Driver Wrapper.....10
6 R&S GTSL Debug Driver Server.....11
7 R&S GTSL Debug Panels.....12
8 R&S GTSL Server Service Control.....20
Glossary: R&S GTSL.....22

1 Documentation Overview

The Debug Driver Server Software is part of the Generic Test Software Library R&S GTSL. For this reason, the following documents are useful in addition to this manual:

- R&S GTSL Generic Test Software Library User Manual
- R&S EGTSL Enhanced Generic Test Software Library User Manual
- R&S IC-Check Generic Test Software Library User Manual

2 Software Installation

You can install the Debug Driver Server Package using the installation routine for the Generic Test Software Library R&S GTSL.

For a detailed description of the installation procedure for the Generic Test Software Library R&S GTSL refer to the following document:

- R&S GTSL Generic Test Software Library user manual, chapter "Software Installation".

3 System Overview

3.1 Block Diagram

Figure 3-1 illustrates how the overall R&S GTSL Software is working with an active Debug Driver Server Software.

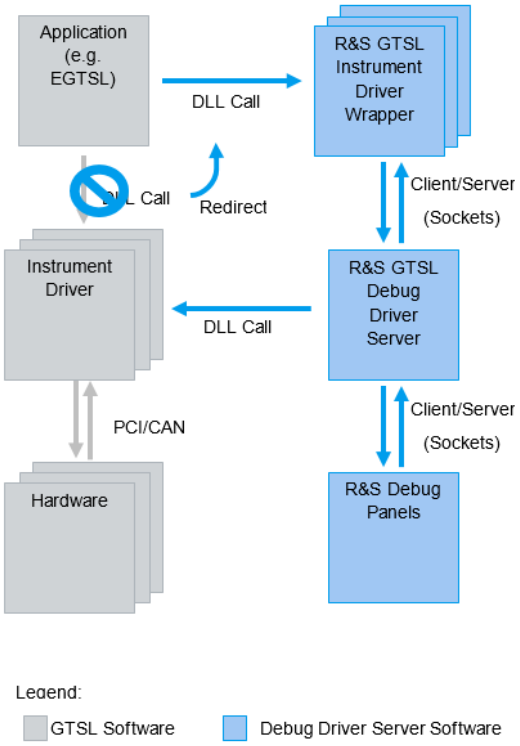


Figure 3-1: System overview

3.2 Information

The R&S GTSL Debug Driver Server Software is considered to give application engineers and circuit test engineers another tool for debugging and implementing actual tests.

Do not use the software in productive systems. Since the commands were redirected over a local server, the runtimes of the messages and the server are added to the actual runtime. The completeness of the messages cannot be ensured, since software from other platforms is involved.

The R&S GTSL Debug Driver Server Software facilitates the information gathering of an ongoing measurement. Since the driver contains no interface like this, an addition

was necessary. The provided application shows additional information even during the actual measurement.

As seen in the block diagram (see Chapter 3.1, "Block Diagram", on page 7), the system ‘hijacks’ the connection to the driver and sends the information to the R&S GTSL Debug Driver Server. Within the server, information from the incoming commands is extracted, executed on the hardware and information is provided to the R&S Debug Panels.

3.3 Available Instrument Drivers

Currently twelve instruments are supported. All other instruments are still working, but the system is not able to monitor those.

- R&S TS-PAM
- R&S TS-PFG
- R&S TS-PIO2
- R&S TS-PIO3B
- R&S TS-PMB
- R&S TS-PSAM
- R&S TS-PSM1
- R&S TS-PSM2
- R&S TS-PSM3
- R&S TS-PSM4
- R&S TS-PSM5
- R&S TS-PSU

4 Quick-Start

The complete R&S GTSL Debug Driver Server Software was developed to be handy and intuitive. Nevertheless, the following chapter consolidates a short guide for setup and running for a simple use case.

1. Activate the R&S GTSL Debug Driver Server Software.
   After the installation of the packages, the system is almost ready for usage. You just have to start the server and enable debugging in the R&S GTSL Debug Panels.
   To open the debug panels, select "File" > "Enable Debugging".

   Note: Enabling debugging is only possible if no R&S GTSL application has loaded any instrument driver. Close all running R&S GTSL applications first.

2. Establish a connection.
   After enabling the system, the instrument driver DLLs within the installation folder of the R&S GTSL software are replaced by the instrument driver wrapper DLLs. Therefore, any application, which uses the binaries in the R&S GTSL installation folder, will automatically use the debug system. Any "init" method (Prefix_init and Prefix_InitWithOptions) of the instrument driver wrapper DLLs will automatically establish a connection to the server.

3. Execute the test application.
   Execute your measurement application as usual. The performance of the system is almost the same. A small overhead is added to every call to the instrument driver wrapper methods.

4. Check information in the R&S GTSL debug panels.
   The information in the application can be refreshed at any time by the controls on the bottom. Nevertheless, it is better to refresh the data during a breakpoint or after a measurement is done. This is because the actual state is mostly unknown, during a measurement.

5. Disable the R&S GTSL Debug Driver Server Software.
   After a successful debugging, when the measurement works as expected, you should disable the system. As already mentioned, there is a small increase in the runtime, which can interfere with the results in the productive system. Therefore, the server should only be active during debugging.

   Note: Disabling debugging is only possible after all previously running R&S GTSL applications have been closed.

5 R&S GTSL Instrument Driver Wrapper

Instead of executing the commands directly on the hardware, the wrapper redirects any input and output data to the R&S GTSL Debug Driver Server. The server executes the command on the hardware and updates its internal state of the hardware. For a convenient usage, the wrapper DLLs replace the actual instrument drivers in the GTSL folder. Therefore, any application which uses the genuine delivered DLLs, works directly with this addition.

After the installation of the R&S GTSL software, the wrapper DLLs are located in `C:\Program Files (x86)\Rohde-Schwarz\GTSL\BinWrapper`. In addition, a second folder `C:\Program Files (x86)\Rohde-Schwarz\GTSL\BinDriver` holds the original instrument driver DLLs for restore purposes. The 64 Bit versions of the wrapper and driver DLLs can be found in the corresponding directories where `Program Files (x86)` is exchanged with `Program Files`. During enabling of the system the R&S GTSL instrument driver wrapper DLLs are copied to the installation folder of the R&S GTSL software and overwrite the actual instrument drivers. To reverse this action and restore the actual behavior again, the backed-up instrument driver DLLs are copied into the GTSL bin folder.

If a user-defined application does not use the installation folder as source or the path variable for the drivers, the debug panels will not work properly.

Do not update firmware while the debug system is active. The wrapper does not support the commands to upload and download the firmware and data stored in the FPGA. Therefore any attempt to update the firmware will not work.

6 R&S GTSL Debug Driver Server

General

The purpose of the server is to accept incoming connections from the wrapper and execute commands with the actual drivers on the hardware. As mentioned already, most of the drivers do not support the extraction of the current state from the hardware. To overcome this issue, the server now achieves the single access to the hardware and is able to manage multiple client connections.

Generally, the communication is based on simple TCP sockets opened by the server. The TCP port (default: 35102) can be configured in the settings, to avoid conflict with other services on the machine.

Note that the software will store information and settings into the registry. This is due to the share status and different information with all involved software modules, for example the wrapper binaries. The used registry folder is already established by the GTSL installation:

32Bit: `HKLM\SOFTWARE\Rohde&Schwarz\GTSL\DebugServer`

64Bit: `HKLM\SOFTWARE\WOW6432Node\Rohde&Schwarz\GTSL\DebugServer`

Service implementation

For a more convenient usage, the server is implemented as a Microsoft Windows service. Effort was put in, to create the tool R&S GTSL Server Service Control (see Chapter 8, "R&S GTSL Server Service Control", on page 20). Although the tool has a small GUI, the complete functionality is also available via the R&S GTSL Debug Panels.

User interactions with the server are reduced to a minimum. After installing and starting the server initially, no further action must be performed until the debug mechanism should be disabled again. The system is activated even after a system reboot, because the service is installed automatically as an automatic start service. Since the server is the central point of the system, it is necessary to run it before any of the actions to be tracked are executed.

The server produces a log file with errors and rudimentary information under `C:\ProgramData\Rohde-Schwarz\GTSL\Logs\service_out.txt`. If the connections from the panel fail to establish, a first hint can be taken from this file. Old log files are overwritten, as soon as the server starts.

7 R&S GTSL Debug Panels

7.1 General

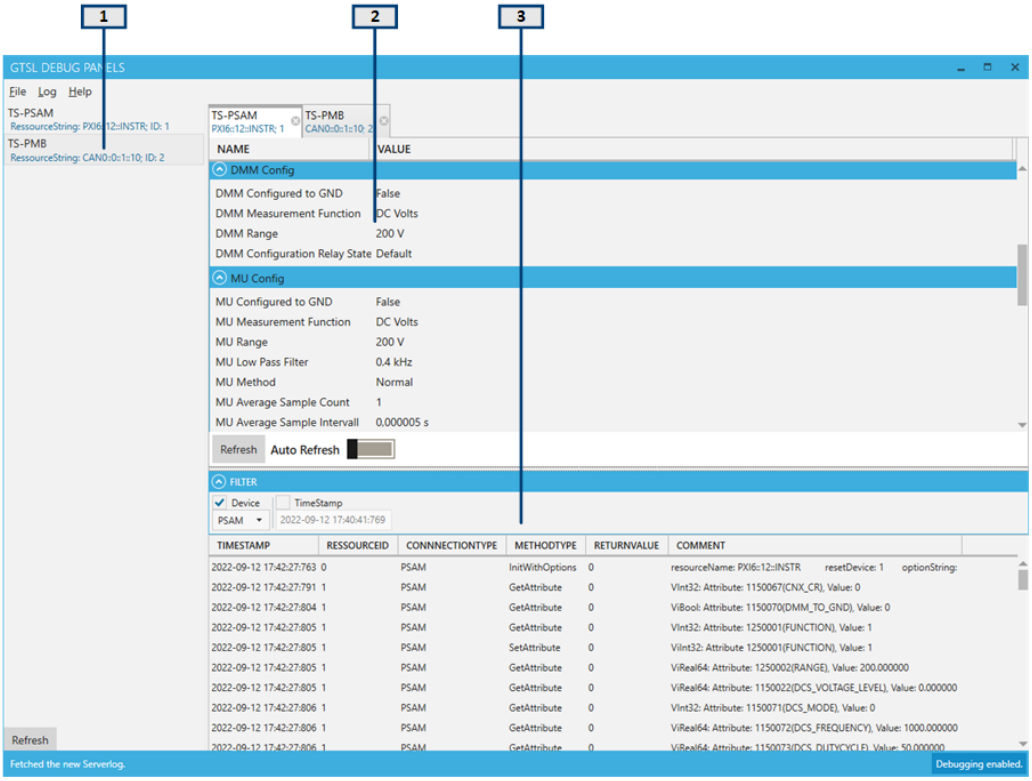
During the startup checks, the global state of the debug system is gathered. If the R&S GTSL Debug Driver Server is not running already, it starts automatically in the background as system service. At the right side of the R&S Debug Panels status bar is shown whether debugging is enabled or disabled at the moment.

Since the R&S GTSL Debug Panels application has to write into the Microsoft Windows Program Files directory and to install and start, respectively to stop and deinstall a Microsoft Windows system service, it is started automatically with administrator privileges.

Common for the current cards is the display of connections to the analog bus to find established connections between single cards. Additionally, the driver version and firmware versions are shown at the very end of the list. Other information is chosen individually for every card.

The R&S GTSL Debug Panels application has a simplified layout and control design to reduce user interaction, since the main purpose for the application is to display information. In fact, the only way to interact with the panel is via two controls on the bottom. The "Refresh" button allows refreshing the information on the active panel once, whereby the "Auto Refresh" switch activates a timer with a tick-rate of one second, which refreshes the panel automatically.

7.2 Main Screen



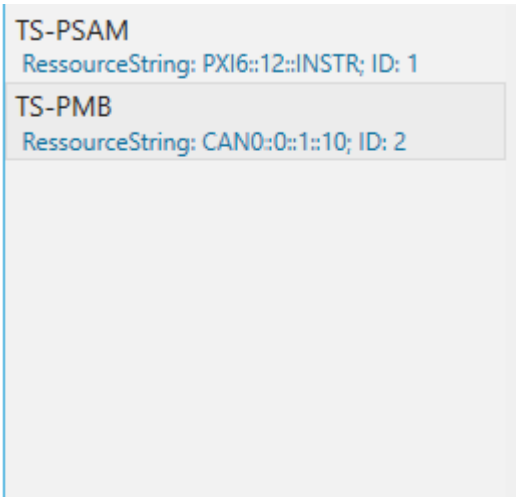
1 = Device list
2 = Main area (several tabs)
3 = Log

The application is divided into three areas, which focus on different functionalities.

- The device list shows all currently initialized instrument drivers on the server.
- The main area contains the actual panels.
- The log shows the protocol after it was requested.

For convenience, drag-able spacers divide those three areas to change the size of the areas. On the bottom, there is also a status bar, which shows various useful information of ongoing processes.

7.2.1 Device List

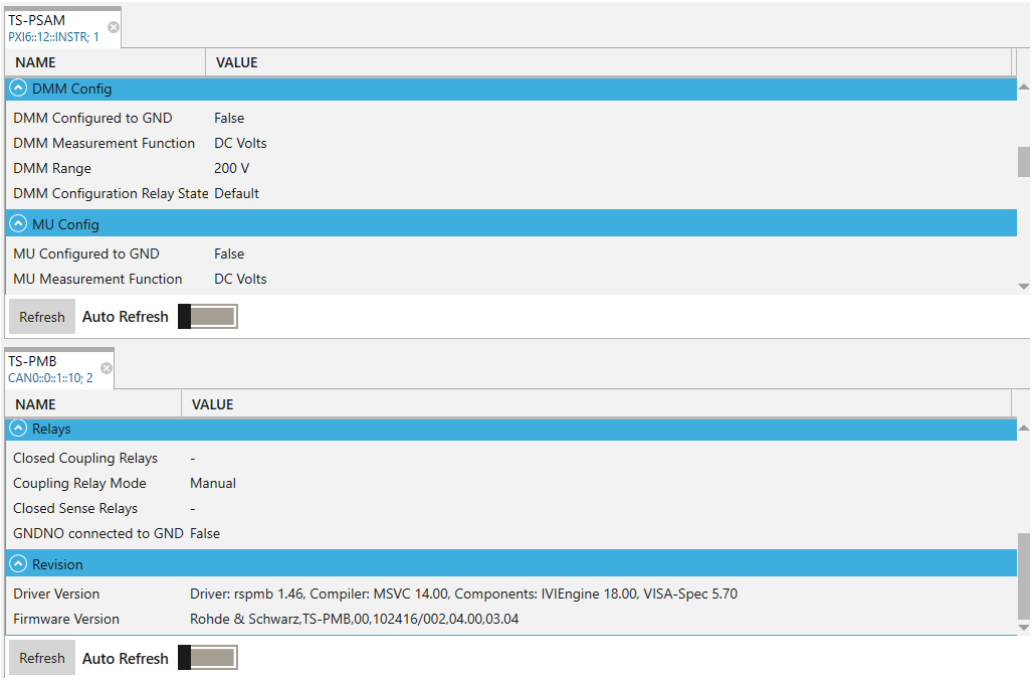


The list on the left holds current initialized devices of the server. Since the system does not support all devices (see Chapter 3.3, "Available Instrument Drivers", on page 8), the list only shows the implemented devices. The blue text underneath the TS module name, shows the actual resource string that you can use to distinguish between different modules of the same type.

Double-clicking on one of those entries selects the corresponding panel. The panel is then shown in the current window. If the selected panel is already opened, the tab will be selected and therefore highlighted in the application.

The "Refresh" button requests the connection list from the server. Already opened device tabs do not close, even if the device is closed on the server already. In addition, you can also use this button, if the connection to the server is closed unexpectedly.

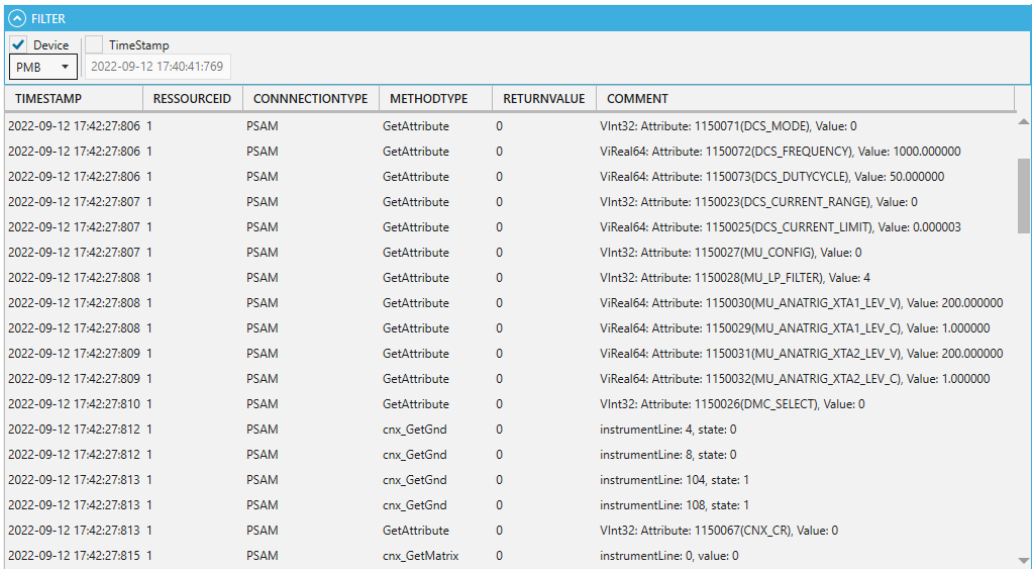
7.2.2 Panel



The panels have the same interface layout for an easy comparison between multiple cards. Furthermore, this also allows the tracking of a switched path over the analog-busses.

The single tabs are drag-able and can therefore be rearranged to get a better suited layout. Additionally, they have a similar behavior to modern tab controller used, for example in IDEs. So a split view is achieved by dragging a tab to any border of a second window.

7.2.3 Log



TIMESTAMP	RESSOURCEID	CONNNECTIONTYPE	METHODTYPE	RETURNVALUE	COMMENT
2022-09-12 17:42:27:806	1	PSAM	GetAttribute	0	VInt32: Attribute: 1150071(DCS_MODE), Value: 0
2022-09-12 17:42:27:806	1	PSAM	GetAttribute	0	ViReal64: Attribute: 1150072(DCS_FREQUENCY), Value: 1000.000000
2022-09-12 17:42:27:806	1	PSAM	GetAttribute	0	ViReal64: Attribute: 1150073(DCS_DUTYCYCLE), Value: 50.000000
2022-09-12 17:42:27:807	1	PSAM	GetAttribute	0	VInt32: Attribute: 1150023(DCS_CURRENT_RANGE), Value: 0
2022-09-12 17:42:27:807	1	PSAM	GetAttribute	0	ViReal64: Attribute: 1150025(DCS_CURRENT_LIMIT), Value: 0.000003
2022-09-12 17:42:27:807	1	PSAM	GetAttribute	0	VInt32: Attribute: 1150027(MU_CONFIG), Value: 0
2022-09-12 17:42:27:808	1	PSAM	GetAttribute	0	VInt32: Attribute: 1150028(MU_LP_FILTER), Value: 4
2022-09-12 17:42:27:808	1	PSAM	GetAttribute	0	ViReal64: Attribute: 1150030(MU_ANATRIG_XTA1_LEV_V), Value: 200.000000
2022-09-12 17:42:27:808	1	PSAM	GetAttribute	0	ViReal64: Attribute: 1150029(MU_ANATRIG_XTA1_LEV_C), Value: 1.000000
2022-09-12 17:42:27:809	1	PSAM	GetAttribute	0	ViReal64: Attribute: 1150031(MU_ANATRIG_XTA2_LEV_V), Value: 200.000000
2022-09-12 17:42:27:809	1	PSAM	GetAttribute	0	ViReal64: Attribute: 1150032(MU_ANATRIG_XTA2_LEV_C), Value: 1.000000
2022-09-12 17:42:27:810	1	PSAM	GetAttribute	0	VInt32: Attribute: 1150026(DMC_SELECT), Value: 0
2022-09-12 17:42:27:812	1	PSAM	cnx_GetGnd	0	instrumentLine: 4, state: 0
2022-09-12 17:42:27:812	1	PSAM	cnx_GetGnd	0	instrumentLine: 8, state: 0
2022-09-12 17:42:27:813	1	PSAM	cnx_GetGnd	0	instrumentLine: 104, state: 1
2022-09-12 17:42:27:813	1	PSAM	cnx_GetGnd	0	instrumentLine: 108, state: 1
2022-09-12 17:42:27:813	1	PSAM	GetAttribute	0	VInt32: Attribute: 1150067(CNX_CR), Value: 0
2022-09-12 17:42:27:815	1	PSAM	cnx_GetMatrix	0	instrumentLine: 0, value: 0

This control shows a log of all traceable actions done by the R&S GTSL Debug Driver Server. During debugging of a test program or other application, the actual sequence of actions provide additional information. Such an entry consists of the timestamp, the actual driver with resource number, as well as a comment, where most of information about the parameter is given. In cohesion with the actual content in the panel, this covers a large percentage of the achievable status.

By design, the actual log resides on the server. Therefore, the displayed protocol is actually a copy. This allows gathering the complete log, even if the R&S Debug Panels were started delayed.

Best practice is to display the log after a sequence of interest, since during the request all actions on the server are delayed.

Besides the possibility to sorting the list by any column, invoked by clicking onto one of the headers, there are two filters. One filter is able to filter the list by devices, which occur at least once in the log. The other filter enables old entries to sort out. All entries with an older timestamp than the given one, are hidden from the user interface.

7.3 Menu Structure

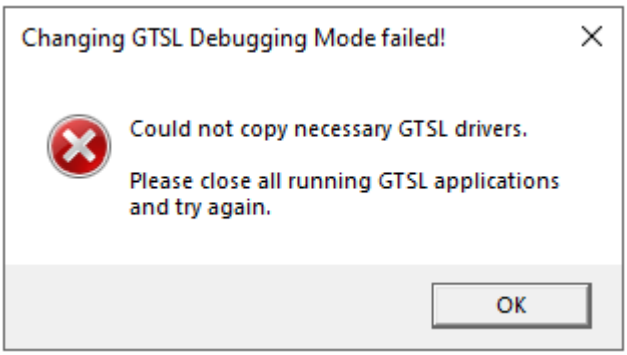
7.3.1 File

Enable Debugging

With the already established system including standalone drivers and several custom-developed applications, introducing the R&S GTSL Debug Driver Server Software is

invasive. This is due to the only interface to the hardware is given on file level by the drivers. To get this work, files have to be replaced in known directories.

To activate the debugging, the R&S GTSL Server Service Control tool overwrites the actual instrument driver with stored wrapper DLLs. If an active program currently has loaded any of the drivers, Windows is not able to overwrite the files and will throw an error. This error is caught by the tool and asks the user to close all running R&S GTSL applications first an then to try again to enable debugging.



After successfully replacing all R&S GTSL Instrument Drivers with Wrapper DLLs, the panel connects to the server and resets the list of devices.

Disable Debugging

Contrary to "Enable Debugging", "Disable Debugging" overwrites all Wrapper DLLs with the original R&S GTSL Instrument Driver binaries again and sets the status bar to "Debugging disabled". Just like in the enable debugging process, disabling debugging is only possible if all running R&S GTSL applications have been closed. None of the DLLs to be replaced must be in use by any application.

While debugging a customer developed application that uses R&S GTSL device drivers with an Integrated Development Environment, e.g. Visual Studio, it will often happen that the debugging process is stopped unexpectedly due to a problem in the code or by simply stopping the Visual Studio Debugger while the program has been halted at a breakpoint.

In this case, the R&S GTSL device driver hasn't been closed properly with its `<drivername>_close()` function. Therefore this device driver still appears in the list of all active devices in the Debug Panels Device List even after refreshing it. To get rid of this obsolete Device List Entry, disable debugging and enable debugging again via the R&S GTSL Debug Panels menu. This resets the list of devices.

Settings

The "System" settings configure the local system. The server port is adjustable, if an already established software on the computer uses the default TCP Port 35102. This setting is read by every module, which participates in the connection. Every wrapper has read access on this variable and establish a connection accordingly to this option.

Reconfigure the TCP Port, if the port is occupied otherwise. All software modules will use the new set port.

Additionally, the log level for the server is configurable. This has an effect on the level of information given in the event log of the server saved in the logfile (see Chapter 7.2.3, "Log", on page 16). However, this does not affect the entries created by methods of the single driver, are stored independent of the level.

The last three points in the system settings determine directories where the software components are stored. 'GTSL Bin Folder' state the binary folder within the installation folder of the GTSL/EGTSL software. This is the target folder for any copy action during enabling and disabling of the system. The other two folders annotate the folder, where copies of the actual drivers and the wrappers are stored respectively.

Changes in the system settings are only applied into the system, after the "Apply" button was clicked. This prevents false configurations, if settings are done accidentally. If the settings window is closed without selecting the "Apply" button, the changes are discarded.

Exit

If the R&S Debug Panels application is closed while debugging is disabled, the R&S GTSL Debug Driver Server which was running in the background also closes and its Windows service will be deactivated.

If the R&S Debug Panels application is closed while debugging is enabled, R&S GTSL Debug Driver Server operations are not affected and continue running.

7.3.2 Log

Request Server-Log

"Request Server-Log" requests the current log from the server. This log holds all actions initialized by the monitored drivers on the hardware. Resulting in a complete history of executed commands, even those events before the panel application started. The gathering of the log stalls the server for a few moments, depending on how many entries the log consists of. During this time, no action from any instrument driver wrapper can be performed. Be aware of this and request the log preferably after a critical situation.

The complete log is shown in the log view on the bottom side of the R&S GTSL Debug Panels after the request was successful. Since this is a server sided log, the entries cannot be reset by restarting the application. A request will gather the complete server log again.

Save Server-Log

After requesting and gathering the server log, those entries can also be saved in a text file. After selecting the "Save Server-Log" entry in the menu, a new dialog will open to choose the actual file name for saving. This can be interesting for different analysis after a debugging session or to compare it with other runs of the same sequence. The whole log is stored to a file.

Clear Server-Log

Clears all server log data and empties the log area of the application.

7.3.3 Help

The "Help" button will open a window, which shows some information about the version and copyright details.

8 R&S GTSL Server Service Control

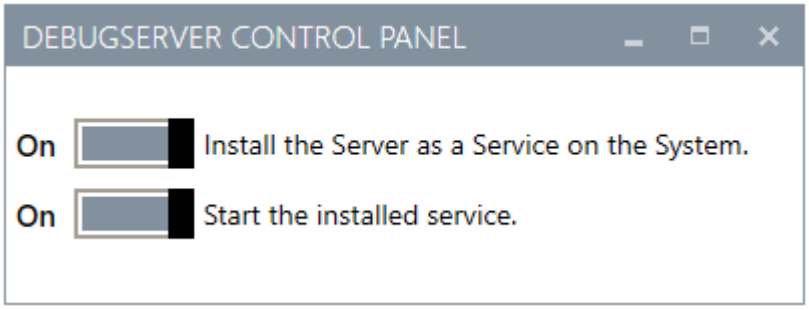
8.1 General

This tool controls the R&S Debug Driver Server Service and enables / disables the debugging system on the computer. The R&S Debug Panels application uses this tool to configure the debugging system via its menu items "Enable Debugging" and "Disable Debugging". Usually there is no need to start this tool directly. Nevertheless the R&S Server Service Control tool provides a very simple user interface where information about the actual status of the debug server and the corresponding Microsoft Windows service can be seen. Each time the tool gets the focus, it updates the status of the switches.

Installing and starting the debug driver server service elevates the user access level and therefore this tool asks for administrator privileges at starttime.

8.2 GUI

The controls act as a display for the actual state and as control to change the described functionality. Use a single click to control the switches. As soon as the switch has changed its state, the actual process is finished in the background.



Install Service

Installs or uninstalls the R&S GTSL Debug Driver Server Service as a Microsoft Windows service. Status "On" means, the service is currently installed. Status "Off" means the service is not installed.

Start Service

The service can be started or stopped. "On" means, the service is started and therefore the R&S GTSL Debug Driver Server is running. "Off" means, the debug server is inactive.

8.3 Command-Line Parameter

The tool can also be controlled via command-line parameters. The following list shows which parameter results in which action from the tool. Be aware that the additional mechanisms are still active. The tool shows a warning if the copying could not be completely done.

Parameter	Action
`-Enablesystem=[ON|OFF]`	Replaces instrument driver DLLs by wrapper DLLs (if "ON") or vice versa (if "OFF")
`-InstallService=[ON|OFF]`	Install (if "ON") or uninstalls (if "OFF") the service. After installation the service may not be started automatically.
`-StartService=[ON|OFF]`	Starts (if "ON") or stops (if "OFF") the service.
`-StartAll`	Fulfill a complete enabling of the system. This option also respects the actual order of those three methods. (1. Enable, 2. Install, 3. Start)
`-StopAll`	Fulfill a complete disabling of the system. This option also respects the actual order of those three methods. (1. Stop, 2. Uninstall, 3. Disabling)

Except `StartAll` and `StopAll` the options can also be combined. For example `ServerServiceControl.exe -InstallService=ON -StartService=ON` installs the service and starts it afterwards.

Glossary: R&S GTSL

D

Debug Driver Server: Central running service which accepts connections from the Wrapper and handles also the connection to the R&S Debug Panels.

Debug Driver System: The name for the collection of the Instrument Driver Wrapper, the Debug Driver Server, the R&S Debug Panels and the Server Service Control tool.

Debug Panels: Application to display the gathered information and control the Debug Driver System.

I

Instrument Driver: Delivered driver DLLs for the TS- Modules.

Instrument Driver Wrapper: Replacements for the Instrument Driver, which connect to the Debug Driver Server and redirect all of the necessary info. These files replace the original DLL's if the debug system is enabled.

S

Server Service Control: A small tool which is capable of install/start the service and enable/disable the whole Debug Driver System.