

Granite River Labs
KayaQ™ Framework Software for Calibration and Test
Automation
Installation and Quick Start Guide

All product names are trademarks, registered trademarks, or service marks of their respective owners.

The GRL disclaims all warranties and liability for the use of this document and the information contained herein and assumes no responsibility for any errors that may appear in this document, nor does the GRL make a commitment to update the information contained herein.

Contact the GRL office to obtain the latest revision of this document

Questions regarding this document in the GRL may be forwarded to:

Granite River Labs 3500 Thomas Road, Suite A, Santa Clara, CA 95054

Phone: 408-627-7608 Fax: 408-912-1810

E-mail: info@graniteriverlabs.com

<http://www.graniteriverlabs.com>

DISCLAIMER

This document is provided "as is" with no warranties whatsoever, including any warranty of merchantability, no infringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. The GRL disclaims all liability for infringement of proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

All product names are trademarks, registered trademarks, or service marks of their respective owners.

Copyright © 2020 Granite River Labs

TABLE OF CONTENTS

1	INTRODUCTION	9
2	RESOURCE REQUIREMENTS	10
2.1	SOFTWARE REQUIREMENTS.....	10
3	SETTING UP GRL KAYAQ AUTOMATION FRAMEWORK	11
3.1	DOWNLOAD AND INSTALL GRL KAYAQ FRAMEWORK	11
3.2	LAUNCH AND SET UP GRL KAYAQ FRAMEWORK	11
3.2.1	Set Up Python Environment	11
4	USING GRL KAYAQ AUTOMATION FRAMEWORK	13
4.1	OVERVIEW OF GRL KAYAQ FRAMEWORK WORKFLOW	13
4.2	DEFINE A NEW PROJECT.....	13
4.3	SET UP EQUIPMENT	15
4.3.1	Manually Verify Connection to Equipment.....	18
4.4	DEFINE TEST SESSION	18
4.5	OVERRIDE EVENTS.....	19
4.5.1	Override Event with Code	20
4.6	DEFINE CONNECTION DIAGRAM.....	21
4.7	DEFINE PVT CONFIGURATION.....	22
4.8	DEFINE TEST CONDITIONS.....	23
4.9	DEFINE TEST SELECTIONS	25
4.9.1	Define Individual Tests.....	25
4.9.2	Use an Existing Module	27
4.9.3	Define Test Groups.....	27
4.10	DEFINE SETUP CONFIGURATIONS.....	28
4.11	DEFINE COMPLIANCE LIMITS.....	30
4.12	RUN TESTS.....	31
4.13	DEFINE TEST RESULTS AND REPORTS	32
4.13.1	Test Session Information.....	33
4.13.2	Test Summary Table.....	34
4.13.3	Test Results.....	34
4.13.4	Delete Test Results.....	36
5	EXAMPLE FOR WRITING A NEW TEST	37
6	USING GRL VISUAL SCRIPTING TOOL	43
6.1	PROGRAMMING EXAMPLE.....	46
7	USING PREDEFINED MODULES	52

7.1 USING STANDARD MEASUREMENT STEP MODULE.....	52
8 APPENDIX A: SAMPLE PROJECTS	56

List of Figures

Figure 1. Set Up Python Environment.....	12
Figure 2. Add or Edit System Variable.....	12
Figure 3. Select Project Options from Application Menu	13
Figure 4. Enter New Project Details	14
Figure 5. Set Up Equipment Example #1.....	15
Figure 6. Set Up Equipment Example #2.....	15
Figure 7. Set Up Equipment Example #3.....	16
Figure 8. Set Up Equipment Example #4.....	16
Figure 9. Set Up Equipment Example #5.....	17
Figure 10. Set Up Equipment Example #6	17
Figure 11. Successful Equipment Setup Connection Example	17
Figure 12. Manually Verify Equipment Connectivity.....	18
Figure 13. Enter DUT Information Example	18
Figure 14. Enter Test Information Example.....	19
Figure 15. Event Override Screen Example.....	20
Figure 16. Event Override Script Entry Example for Python Code	20
Figure 17. Define Connection Diagram Example #1.....	21
Figure 18. Define Connection Diagram Example #2.....	21
Figure 19. PVT Configuration Example #1	22
Figure 20. PVT Configuration Example #2	22
Figure 21. PVT Configuration Example #3	23
Figure 22. Define Test Conditions Example #1.....	24
Figure 23. Define Test Conditions Example #2.....	24
Figure 24. Define Test Conditions Example #3.....	24
Figure 25. Select Tests Screen	25
Figure 26. Define Tests Example #1.....	25
Figure 27. Define Tests Example #2.....	26
Figure 28. Define Tests Example #3.....	26
Figure 29. Define Tests Example #4 – Predefined Module	27
Figure 30. Define Test Group Example.....	28
Figure 31. Define Setup Configuration– Add Custom Page.....	29
Figure 32. Define Setup Configuration– Add Custom Control #1.....	29
Figure 33. Define Setup Configuration– Add Custom Control #2.....	29
Figure 34. Define Setup Configuration– Add Setup.....	30
Figure 35. Define Setup Configuration– Add Variables in Tree Format.....	30
Figure 36. Define Compliance Limits– Add Conditional Limits Example	31
Figure 37. Define Compliance Limits– Change Conditional Limits Example	31
Figure 38. Run Tests Screen.....	32

Figure 39. Generate Report Screen #1	32
Figure 40. Generate Report Screen #2	33
Figure 41. Test Session Information Example.....	33
Figure 42. Test Summary Table Example	34
Figure 43. Test Results Example.....	35
Figure 44. Test Report Deleted	36
Figure 45. Writing Test Example– Edit Test.....	37
Figure 46. Writing Test Example– Write/Define Test Behavior.....	37
Figure 47. Writing Test Example– Using C# Script #1	38
Figure 48. Writing Test Example– Using C# Script #2	39
Figure 49. Writing Test Example– Using Python Script #1	40
Figure 50. Writing Test Example– Using Python Script #2	40
Figure 51. Writing Test Example– Using Visual Script.....	41
Figure 52. Writing Test Example– Using Predefined Module.....	42
Figure 53. Using GRL Visual Scripting Tool Example– Launch Visual Script Tool #1	43
Figure 54. Using GRL Visual Scripting Tool Example– Launch Visual Script Tool #2	43
Figure 55. Using GRL Visual Scripting Tool Example– Visual Script Tool Features	44
Figure 56. Using GRL Visual Scripting Tool Example– Using Variables	44
Figure 57. Using GRL Visual Scripting Tool Example– Add Node #1	45
Figure 58. Using GRL Visual Scripting Tool Example– Add Node #2.....	45
Figure 59. Using GRL Visual Scripting Tool Example– Perform Debugging and Stepping	46
Figure 60. Using GRL Visual Scripting Tool Example– Select Main Function Groups for Programming	47
Figure 61. Using GRL Visual Scripting Tool Example– Access Main Function Group API's.....	47
Figure 62. Using GRL Visual Scripting Tool Example– Access API Info.....	48
Figure 63. Using GRL Visual Scripting Tool Example– Copy Command Line to Clipboard	48
Figure 64. Using GRL Visual Scripting Tool Example– Fraction Code Command #1	49
Figure 65. Using GRL Visual Scripting Tool Example– Fraction Code Command #2.....	50
Figure 66. Using GRL Visual Scripting Tool Example– Fraction Code Command #3.....	50
Figure 67. Using GRL Visual Scripting Tool Example– Fraction Code Command #4.....	51
Figure 68. Using Predefined Modules Example– Standard Measurement Step Module GUI.....	54
Figure 69. Using Predefined Modules Example– Write Code to Read Scope Measurements.....	55

List of Tables

Table 1. Software Requirements 10

Revision Record

Revision	Date	Description of Changes	Author(s)
1.6	03/2019	New document layout for existing content	Ong Gaik Pheng (GRL) gpong@graniteriverlabs.com
2.0	03/2020	Added Section 6.1, Programming Example	Ong Gaik Pheng (GRL) gpong@graniteriverlabs.com

1 Introduction

This Installation & Quick Start Guide provides information on using the GRL KayaQ™ framework software application to automate receiver (Rx) or transmitter (Tx) testing with various equipment.

The GRL KayaQ software platform provides a single framework for all standards and applications, allowing test engineers to combine scripts and equipment setup files into a test flow to repeat one or more Rx/Tx tests. GRL KayaQ can be used to automate CTS MOI's across multiple Scope/BERT/VNA platforms or integrated into the user's own automation environment. The user can also use GRL KayaQ to extend standard solutions to include other test equipment and test case looping such as RF switches, temperature and voltage control equipment as well as aggregate test reports.

The following applications are supported by the GRL KayaQ framework:

- PHY Transmitter Testing
 - Test Equipment Setup
 - DUT Control for Waveform Capture
 - Scope Setup & Sequencing
 - Reporting
- PHY Receiver Testing
 - Test Equipment Setup
 - SSG Calibration with Oscilloscope – RJ, SJ, ISI, TJ, EH
 - BER Testing Automation – Loopback with ED or DUT Automation
 - Margin Testing
 - Reporting
- PHY Return Loss Testing
 - Test Equipment Setup
 - ENA/VNA Calibration
 - Testing
 - Reporting

The GRL KayaQ framework allows further customization when integrated into most lab setups or testplans. For example, GRL KayaQ supports up to x16 lane switching integration and PVT/stress tests integration. Through GRL KayaQ, the user can also add custom tests using Python or GRL Visual Scripts as well as controlling GRL applications with third party software through API.

For GRL's full KayaQ automation framework license and further details, contact GRL at support@graniteriverlabs.com.

2 Resource Requirements

Note: Below are software requirements for the standard GRL KayaQ test platform. Additional software is required to support scripts of various types if using third party tools.

2.1 Software Requirements

TABLE 1. SOFTWARE REQUIREMENTS

Software	Source
python27	http://www.python.org/download/releases/2.7.6/
Python for .Net	http://sourceforge.net/projects/pythonnet/files/
Visual Studio Express	Microsoft (optional)

3 Setting Up GRL KayaQ Automation Framework

This section provides the procedures to start up and configure the GRL KayaQ framework for operation. It also helps users familiarize themselves with the basic workflow of the framework.

Note: The GRL KayaQ framework software installer will automatically create shortcuts in the Desktop and Start Menu when installing the software.

To start using GRL KayaQ, follow the procedures in the following sections.

3.1 Download and Install GRL KayaQ Framework

1. Download the **GRLFrameworkInstallerXXXX.exe** package from the Granite River Labs support site.
2. Run the installer package on the computer where GRL KayaQ is to be used to install the GRL KayaQ framework application.

3.2 Launch and Set Up GRL KayaQ Framework

1. Once the GRL KayaQ framework application is installed, launch the application and go to License→License Details to enable license. *For Demo and Beta Customer License Keys, please request an Activation Key by contacting support@graniteriverlabs.com. Copy the license string and send to support@graniteriverlabs.com to obtain a license key.*
2. Install python27 from <http://www.python.org/download/releases/2.7.6/>. Make sure the installation path is C:\Python27. *Note: Do not use version Python33.*
3. Install Visual C# if script is to be written in C#.
4. Install VISA components (e.g., TekVISA, Agilent IO).
5. Install other third party components if any (e.g., ScopeApp/RX app RPI module).

Note: The GRL KayaQ framework is an automation platform to ease automation efforts only and does not license Python, Visual C# or any third-party components. Users of the GRL KayaQ framework need to ensure eligibility to use these third party tools.

3.2.1 Set Up Python Environment

1. Go to My Computer → Properties.
2. Select **Advanced Settings** and click on **Environment Variables**.
3. Look for the 'PYTHONPATH' variable in the System variables subpanel. If 'PYTHONPATH' does not exist, then click on the **New...** button; otherwise click **Edit...**

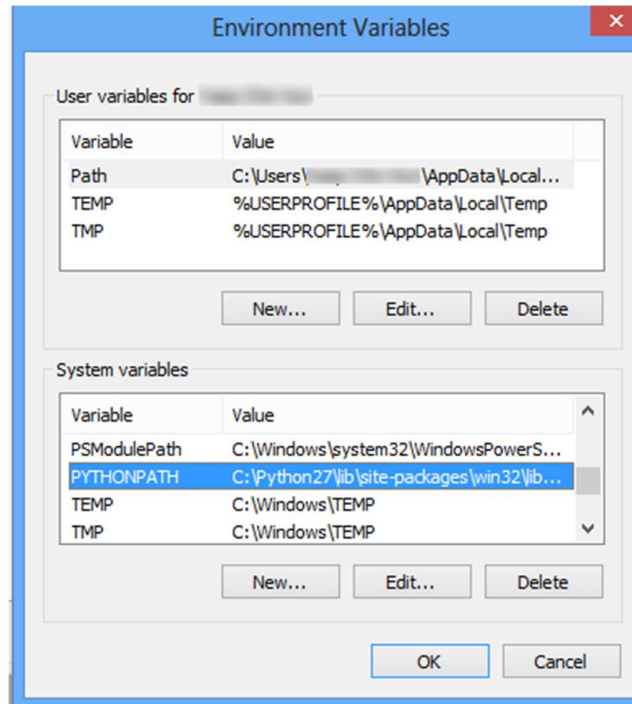


FIGURE 1. SET UP PYTHON ENVIRONMENT

4. Add or edit the system variable as follows:
 - Variable Name: **PYTHONPATH**
 - Variable Value: **C:\GRL\GRLFramework\PythonLib**
5. Add or edit the system variable as follows:

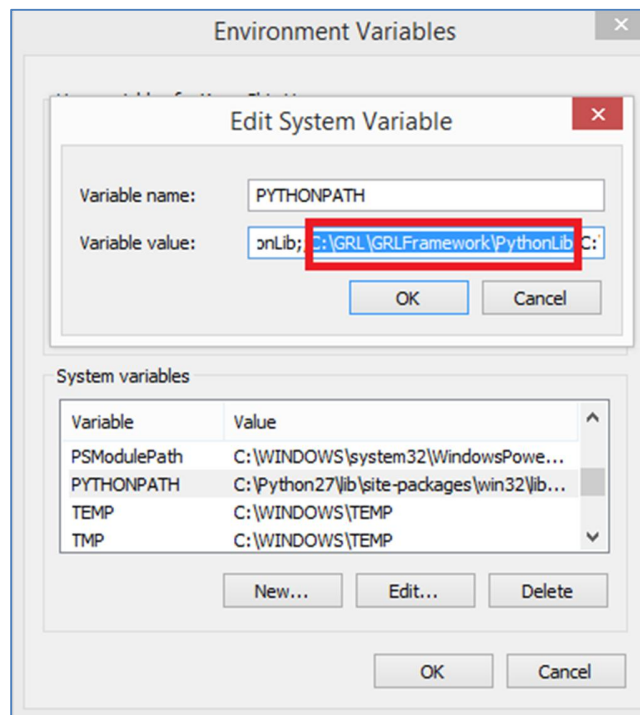


FIGURE 2. ADD OR EDIT SYSTEM VARIABLE

4 Using GRL KayaQ Automation Framework

This section describes how the GRL KayaQ framework can be used to create a new Project, consisting of an equipment setup, a series of tests or test groups and output reports.

4.1 Overview of GRL KayaQ Framework Workflow

Follow these steps to fully define a Project using the GRL KayaQ framework:

- a) Select the GRL KayaQ framework application from the Application menu drop-down list.
- b) Define the Project by name and description.
- c) Define the equipment to be used, including their connection addresses.
- d) Define any “Event Overrides”.
- e) Define a connection diagram for the equipment setup.
- f) Define environmental limits, for iterating a test across ranges.
- g) Define any Test Conditions to be used in one or more tests.
- h) Define the Tests:
 - i. Define each Test.
 - ii. Iterate through all tests in the Test Group.
- i) Create additional Test Groups as needed.
- j) Define one or more Test Reports.

4.2 Define a New Project

Create a new project from the Application tab on the menu bar.

1. Select ‘GRL Framework’ from the Start Menu → GRL.
2. Select the ‘Framework Test Solution’ option from the Application menu drop-down list.
3. On the Application menu, select to start a new Project (Application), Save Project or Load Project.

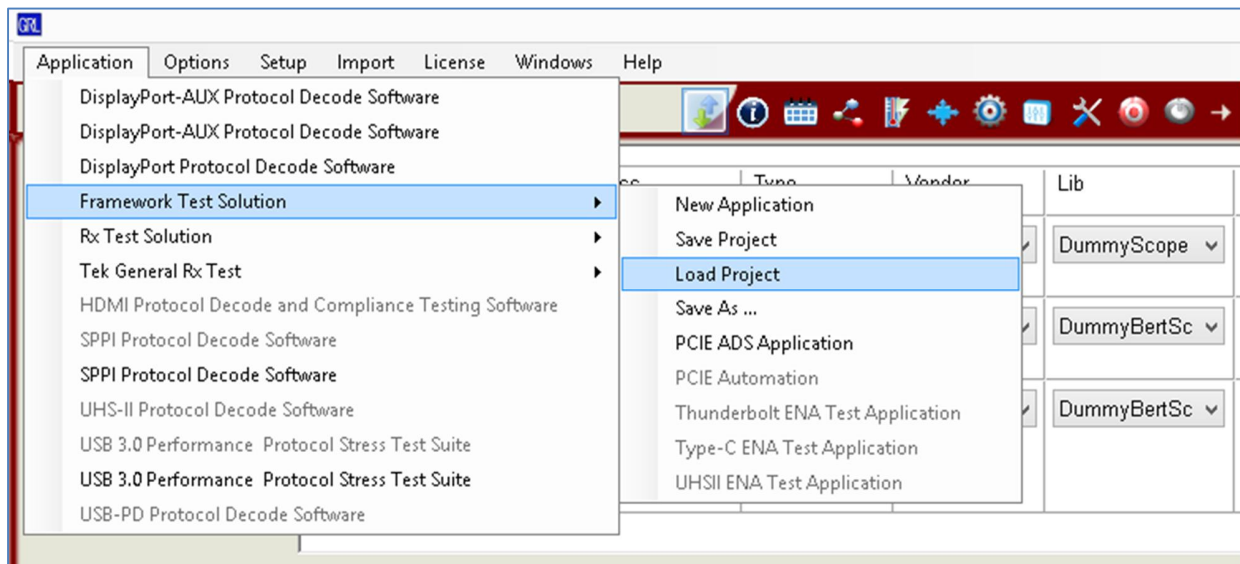


FIGURE 3. SELECT PROJECT OPTIONS FROM APPLICATION MENU

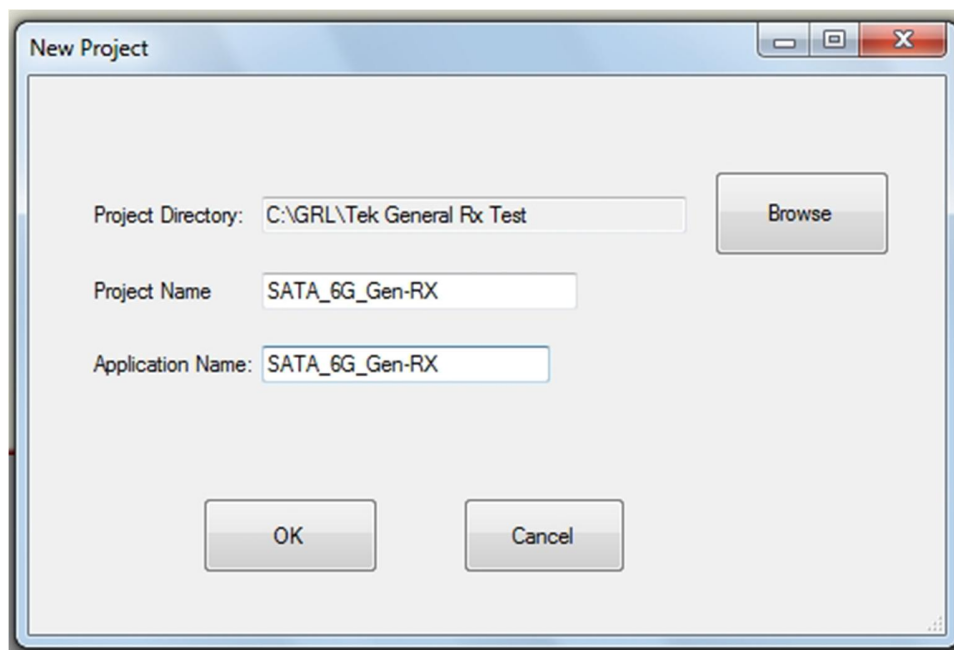
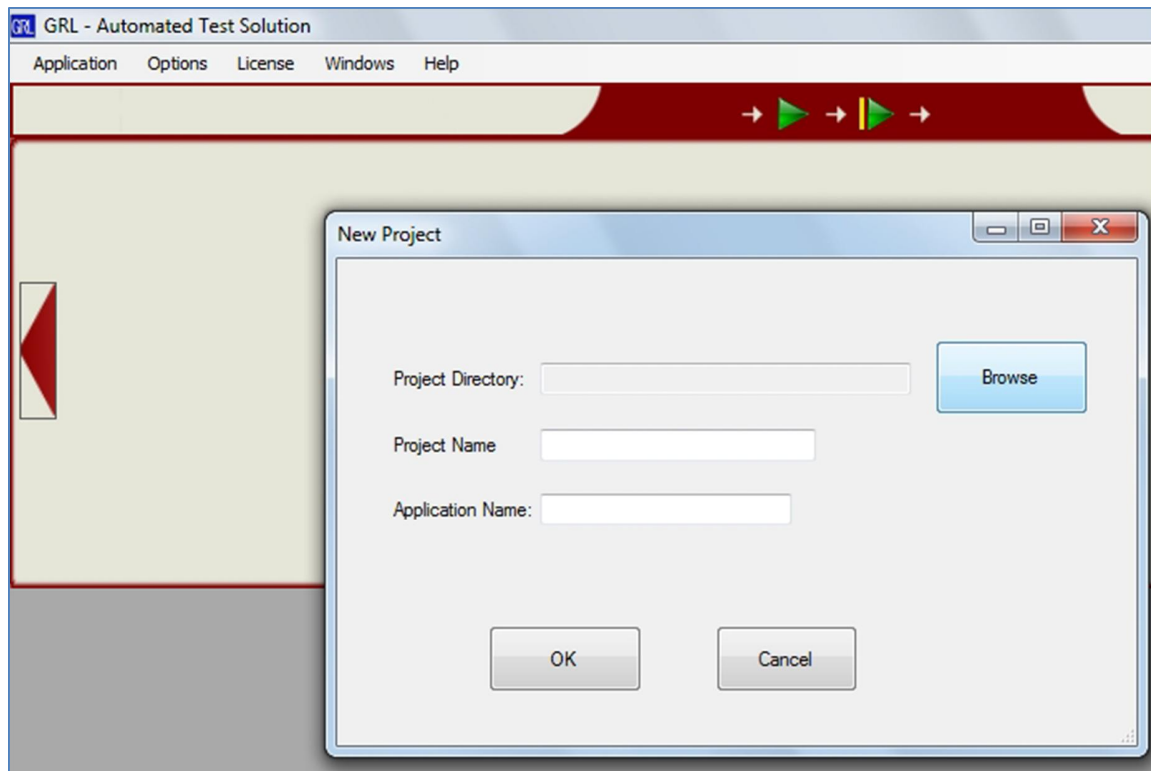


FIGURE 4. ENTER NEW PROJECT DETAILS

4.3 Set Up Equipment

Select the  icon under the main menu to access the Equipment Setup screen.

1. Click on the 'Add Equipment' button on the right to add an equipment.
2. Skip the page if equipment is not required.
3. The assigned ID is used by the software code to access the equipment.
4. Follow the example in the figures below.
5. Repeat the process of entering connection addresses, covering all the relevant equipment.
6. Select the "lightning" button () for each connected equipment, which should turn green () once the software has successfully established connection with each equipment.

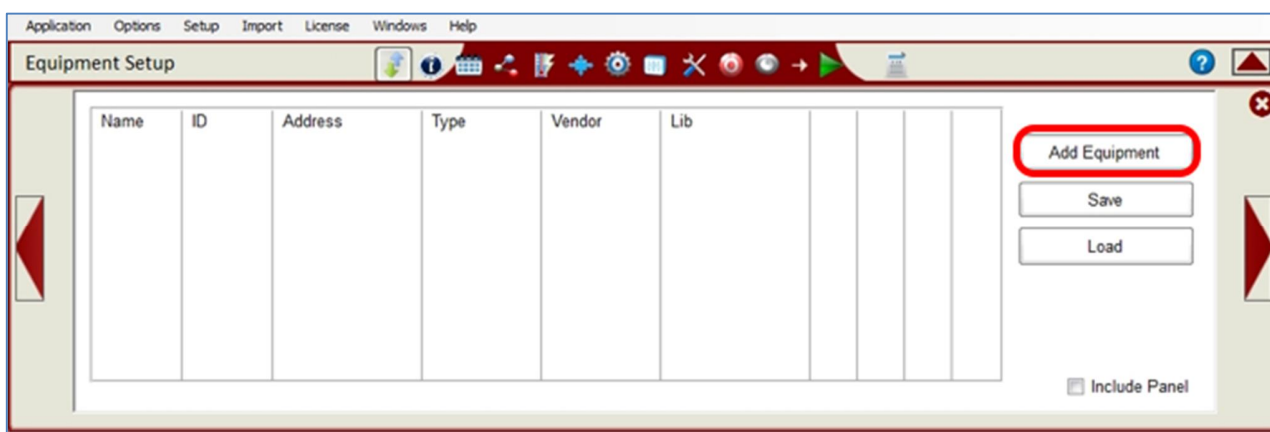


FIGURE 5. SET UP EQUIPMENT EXAMPLE #1

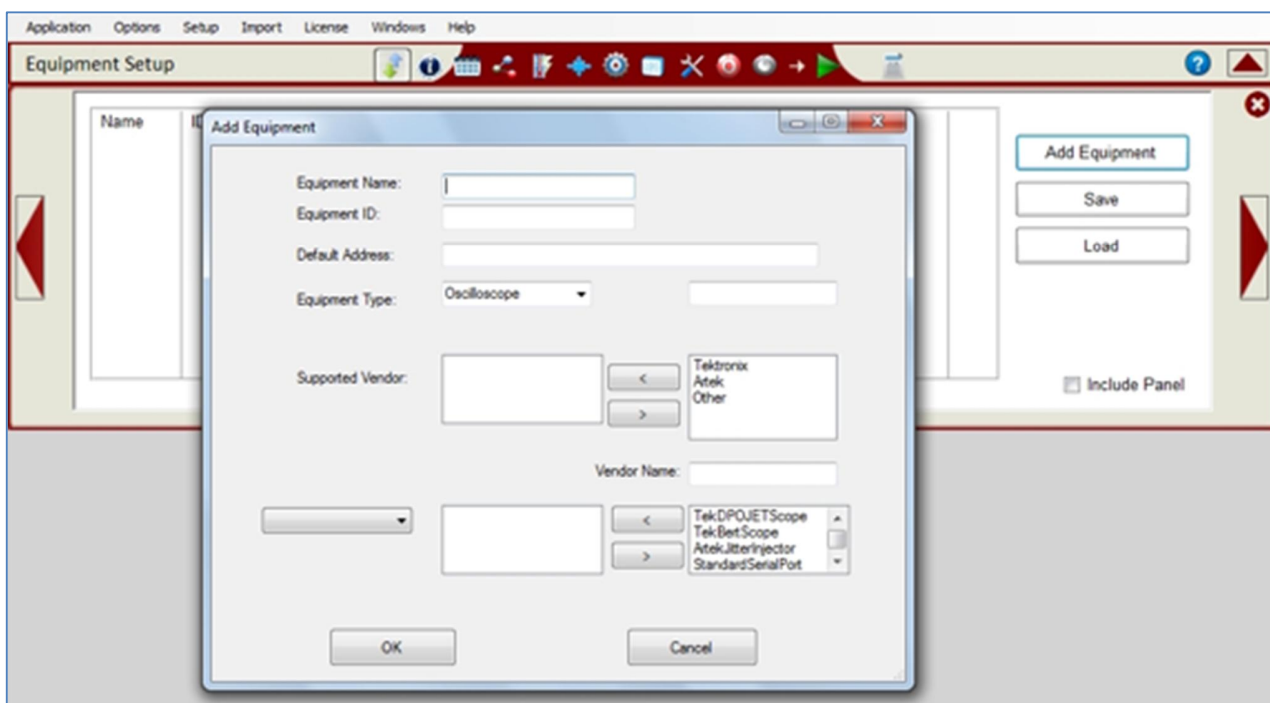


FIGURE 6. SET UP EQUIPMENT EXAMPLE #2

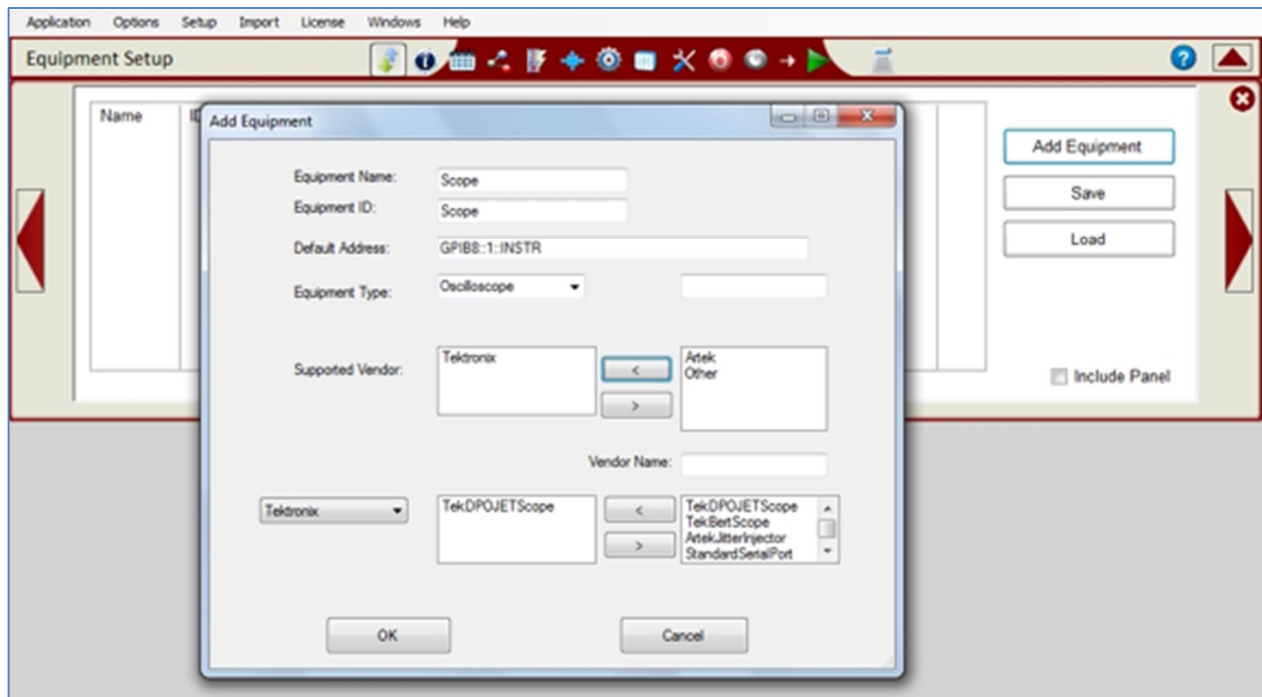


FIGURE 7. SET UP EQUIPMENT EXAMPLE #3

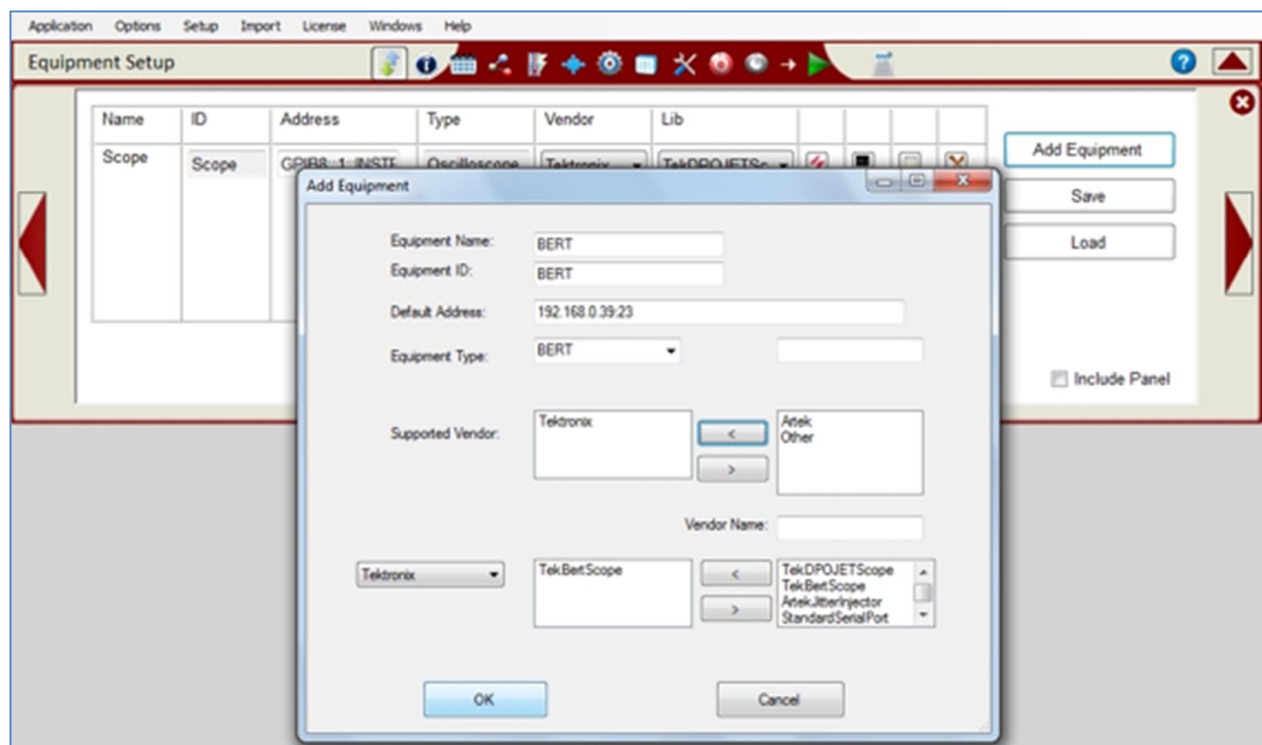


FIGURE 8. SET UP EQUIPMENT EXAMPLE #4

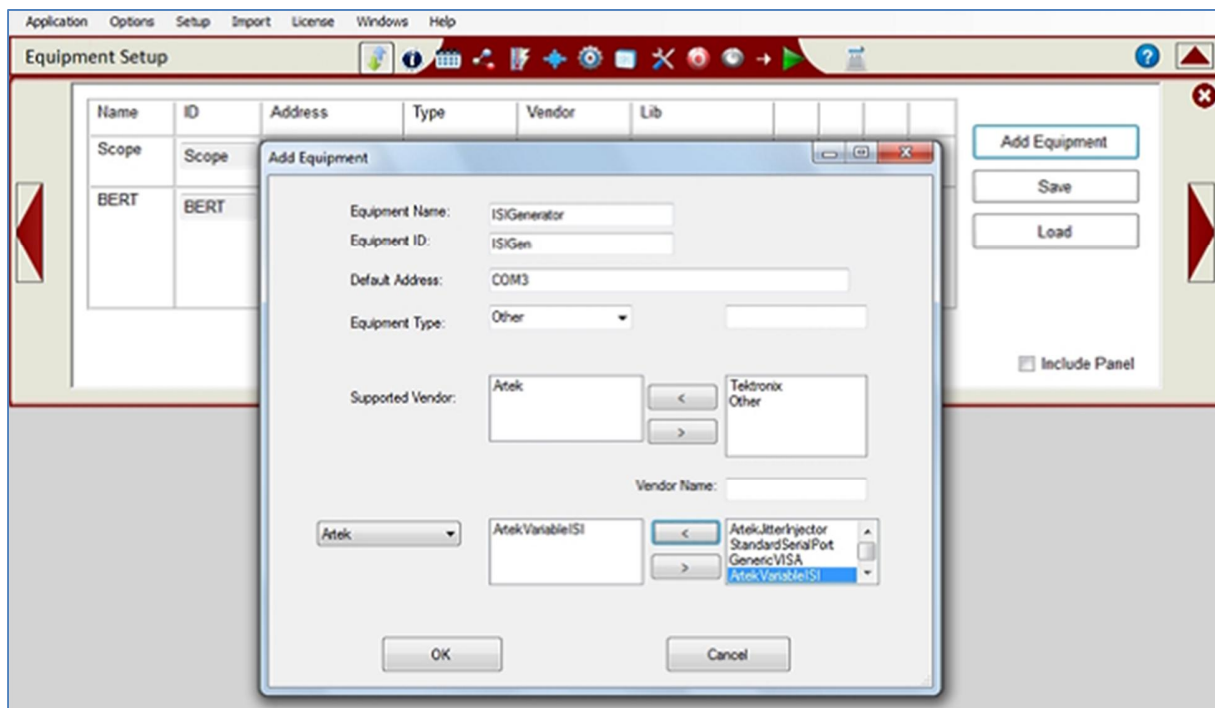


FIGURE 9. SET UP EQUIPMENT EXAMPLE #5

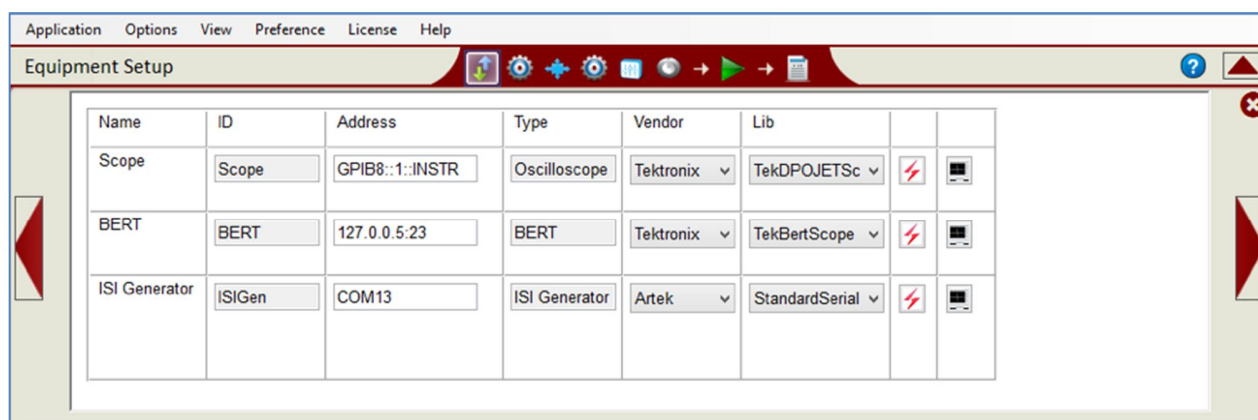


FIGURE 10. SET UP EQUIPMENT EXAMPLE #6

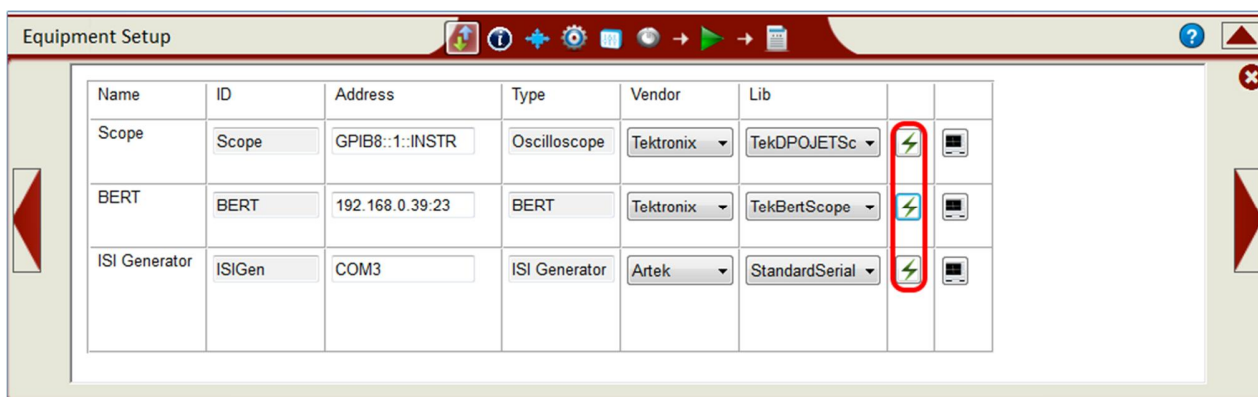


FIGURE 11. SUCCESSFUL EQUIPMENT SETUP CONNECTION EXAMPLE

4.3.1 Manually Verify Connection to Equipment

The developer can also select the  icon (beside the “lightning” icon) to issue a command manually to an equipment. For example, for a GPIB supported equipment, “*idn?” is the recommended way to check if the equipment responds to the framework correctly, as shown below:

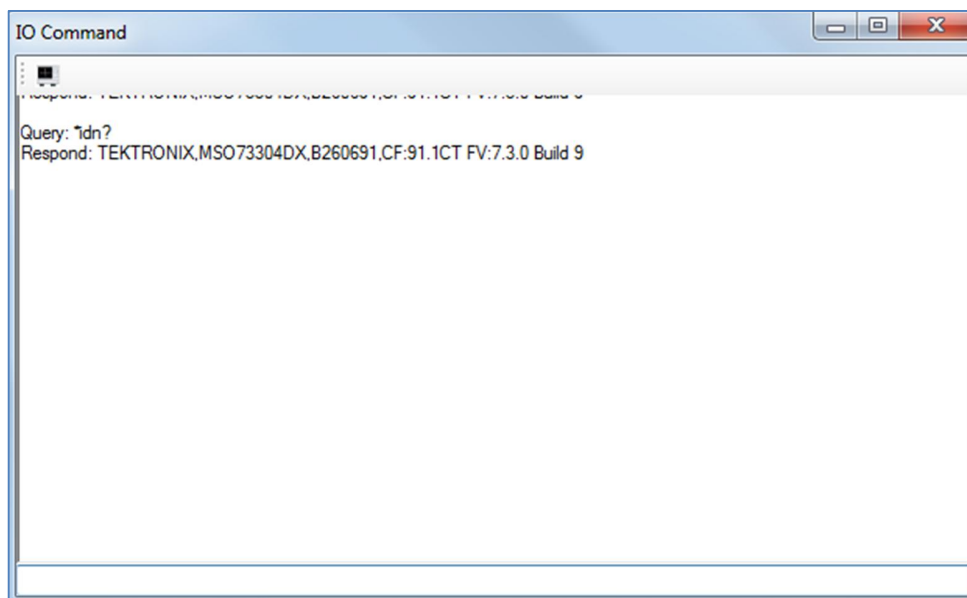


FIGURE 12. MANUALLY VERIFY EQUIPMENT CONNECTIVITY

4.4 Define Test Session

Select the  icon under the main menu to access the Session Info screen which provides information fields for the test session to be run. The information entered will be included in the test report generated by the framework once tests are completed.

- The fields under **DUT Info** and **Test Info** are defined by the user.
- The **Software Info** field is automatically populated by the software.

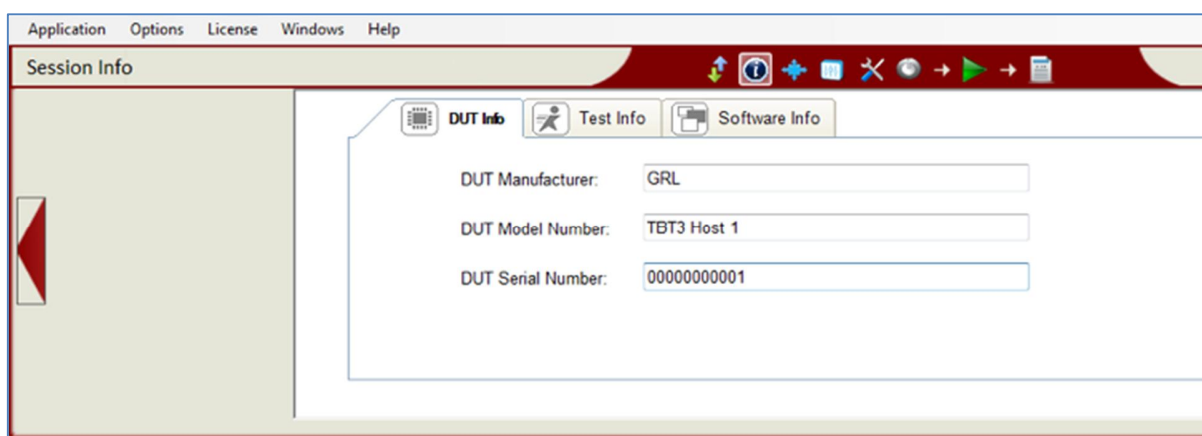


FIGURE 13. ENTER DUT INFORMATION EXAMPLE

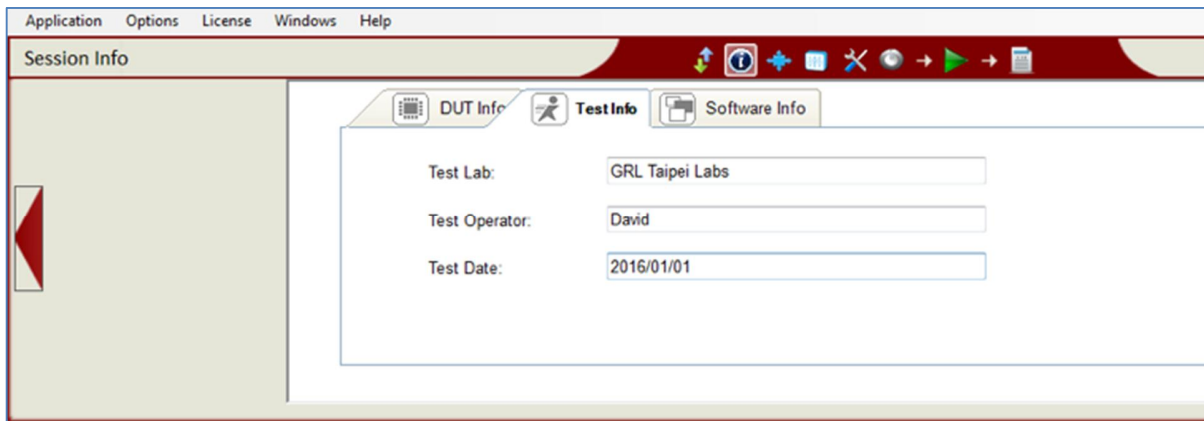


FIGURE 14. ENTER TEST INFORMATION EXAMPLE

4.5 Override Events

The Event Override screen allows the user to override an event with code. The events listed here are supported by the GRL KayaQ framework:

- **OnRunStarted:** An event that is triggered before a new run session is started. For instance, one might initiate the DUT initial state here.
- **OnTestConditionedChanged:** An event that is triggered when any condition that is defined in the 'Condition' screen is called. For instance, this might be useful for one to change a DUT state (e.g., changing the DUT bit rate or control the switch matrix to a different lane).
- **OnStressTestConditionChanged:** An event that is triggered when any condition that is defined in the 'PVT Condition' screen is changed (e.g., changing voltage or temperature).
- **OnTestGroupChanged:** An event that is triggered when "Test group" of test is changed.
- **OnTestStarted:** An event that is triggered before starting a test item.
- **OnRunEnded:** An event that is triggered after a run session is completed. For instance, one might aggregate data in this event.
- **OnUserStopped:** An event that is triggered when a test run is stopped after clicking on the "Stop" button. For instance, one might clean up/reset the DUT state after a test is cancelled.
- **OnControlChanged:** An event that is triggered when a GUI control component (e.g., checkbox, combobox, textbox) defined in the framework receives user input. For instance, once might hide certain tests to be seen by end users when different specs (defined by a text box) are changed.

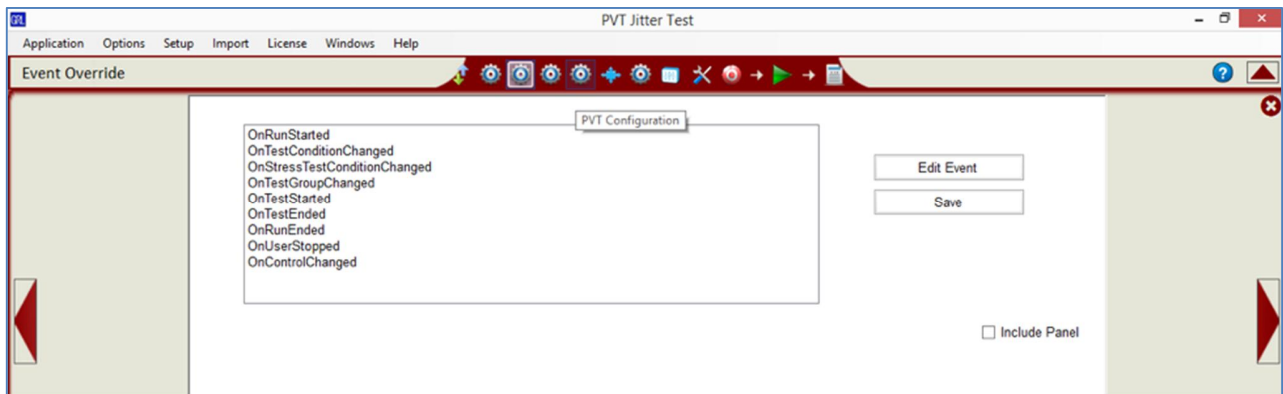


FIGURE 15. EVENT OVERRIDE SCREEN EXAMPLE

4.5.1 Override Event with Code

1. Select any of the events and select **Edit Event**.
2. Select the type of code (e.g., Python, Visual Script) to be used when the 'Event' dialog is opened.
3. Provide a Name for the script and select **Edit Script**. A script will open in a new window for the user to write code to override the event.

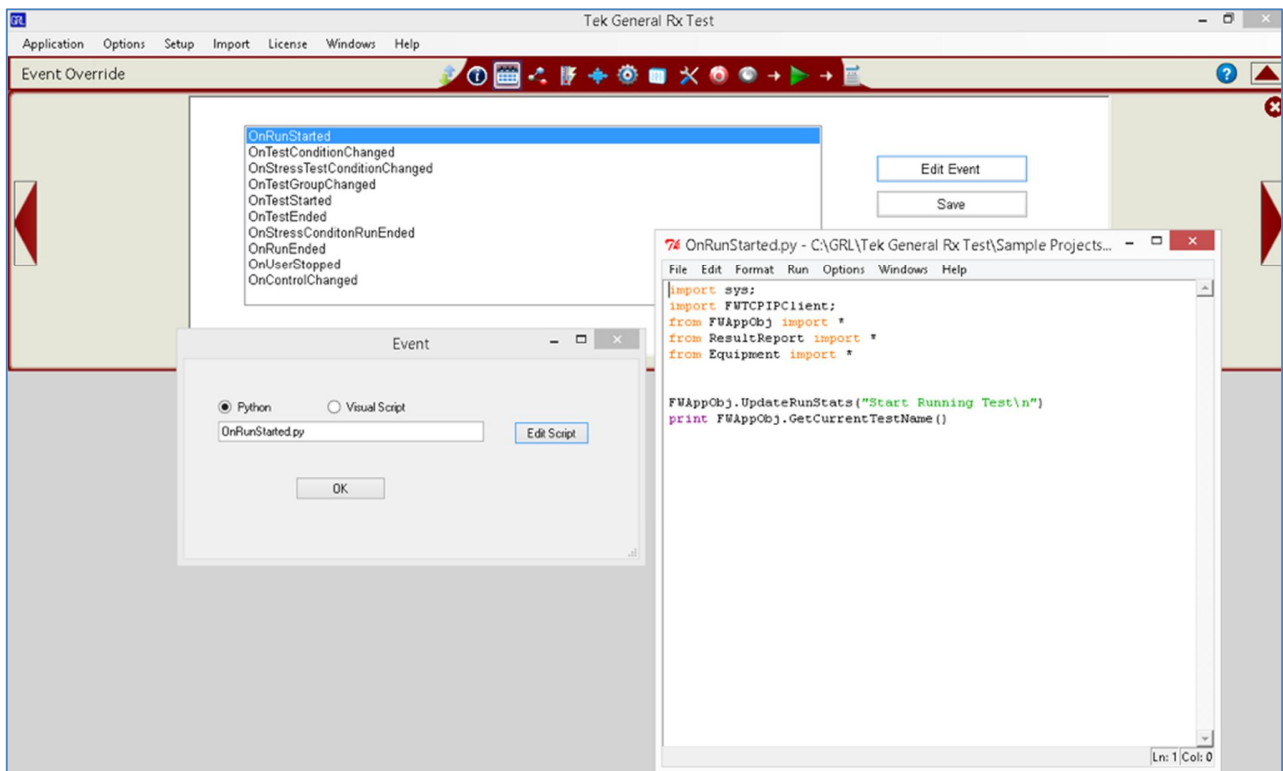


FIGURE 16. EVENT OVERRIDE SCRIPT ENTRY EXAMPLE FOR PYTHON CODE

4.6 Define Connection Diagram

The suite of tests is usually documented with a diagram showing the equipment setup. This helps the user to properly connect the devices, and to proceed with the tests.

The Connection Diagram Maker screen allows the user to add a connection diagram and give a unique name to it. The connection diagram can then be tied to the test created later in the 'Select Tests' screen.

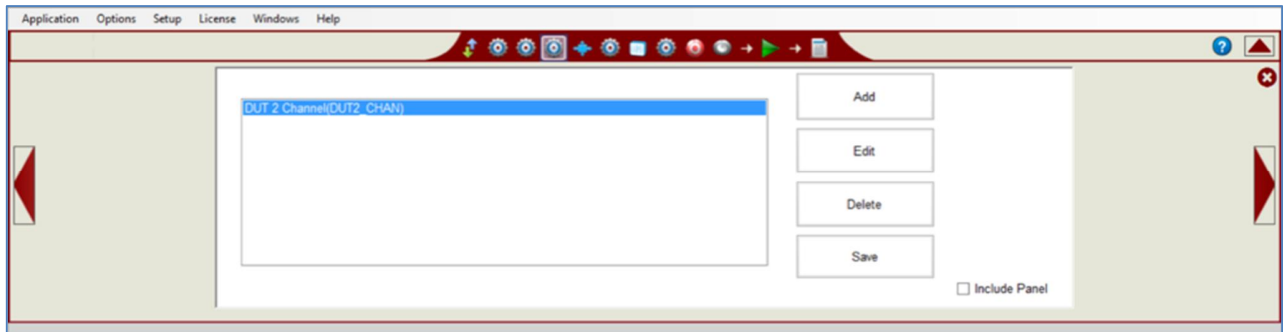


FIGURE 17. DEFINE CONNECTION DIAGRAM EXAMPLE #1

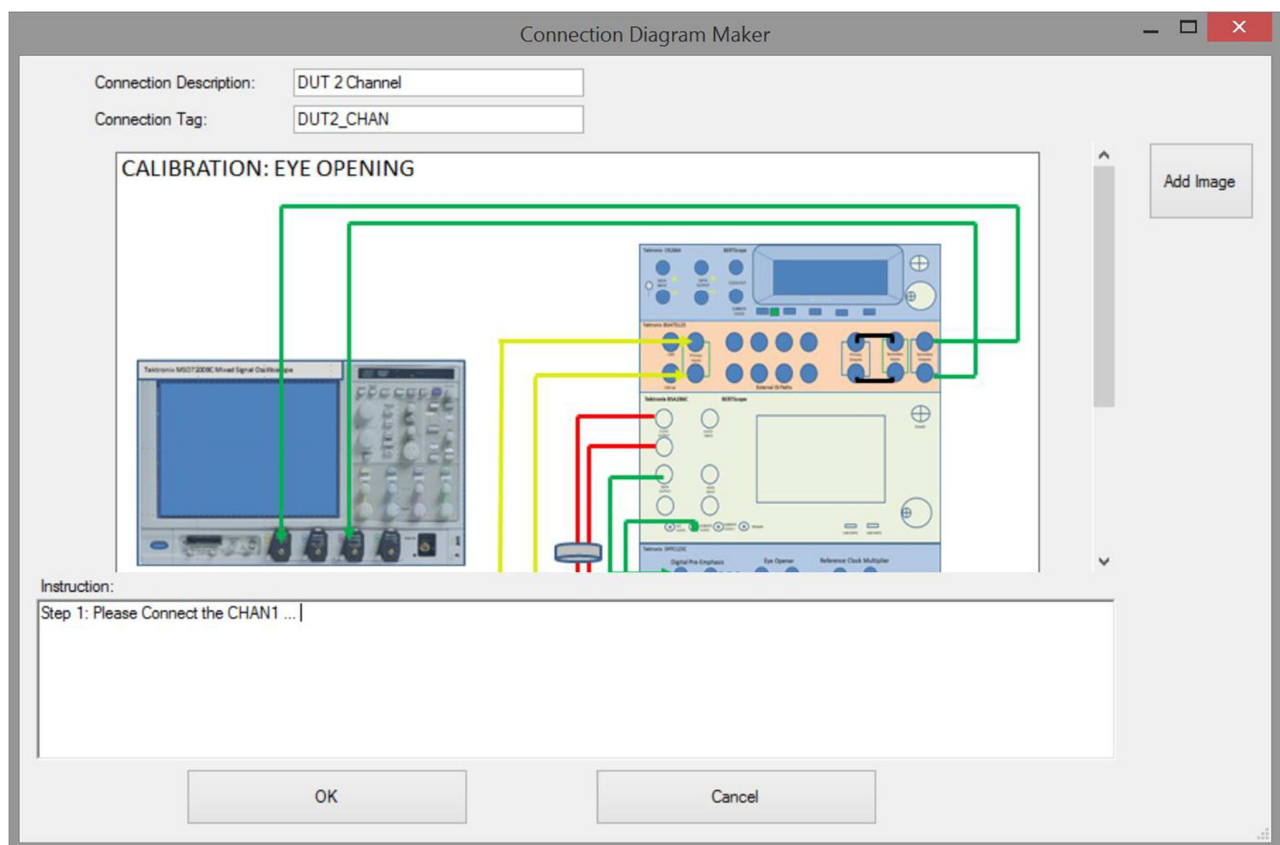


FIGURE 18. DEFINE CONNECTION DIAGRAM EXAMPLE #2

4.7 Define PVT Configuration

The 'PVT Configuration' screen integrates PVT capability in such a way that the user only needs to define the PVT conditions, and a PVT test plan will be automatically populated for all the written tests.

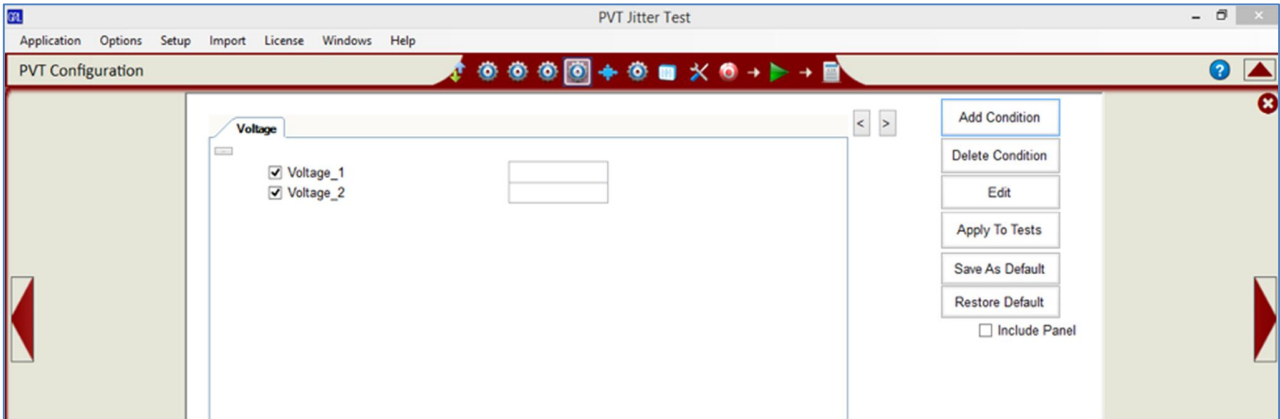


FIGURE 19. PVT CONFIGURATION EXAMPLE #1

To add a PVT condition:

1. Click on the **Add Condition** button on the right and a dialog will pop up as below.
2. In the example below, definitions are made for 3 corners– T1, T2 and T3, and the default settings for T1, T2, T3 are 10C, 20C, 30C, respectively.

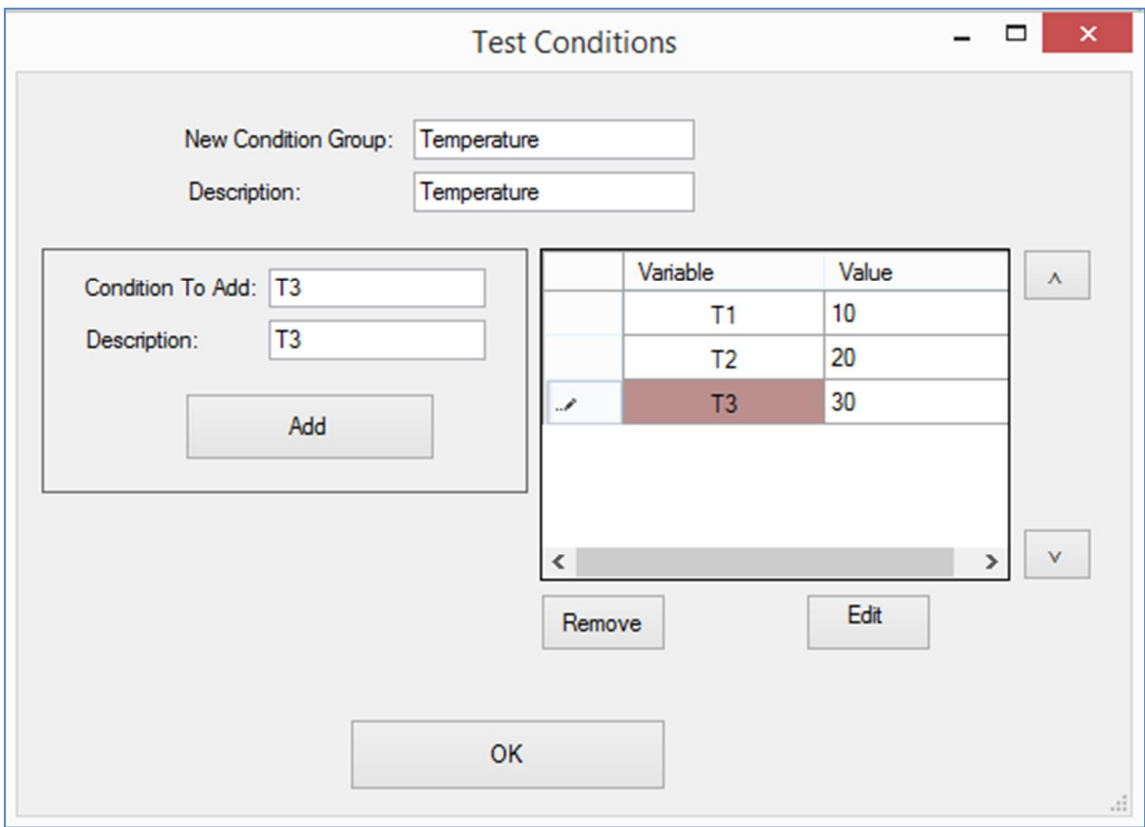


FIGURE 20. PVT CONFIGURATION EXAMPLE #2

3. Click **OK** after defining the PVT condition. Now a new column of conditions is added to the “PVT Configuration” page as below.

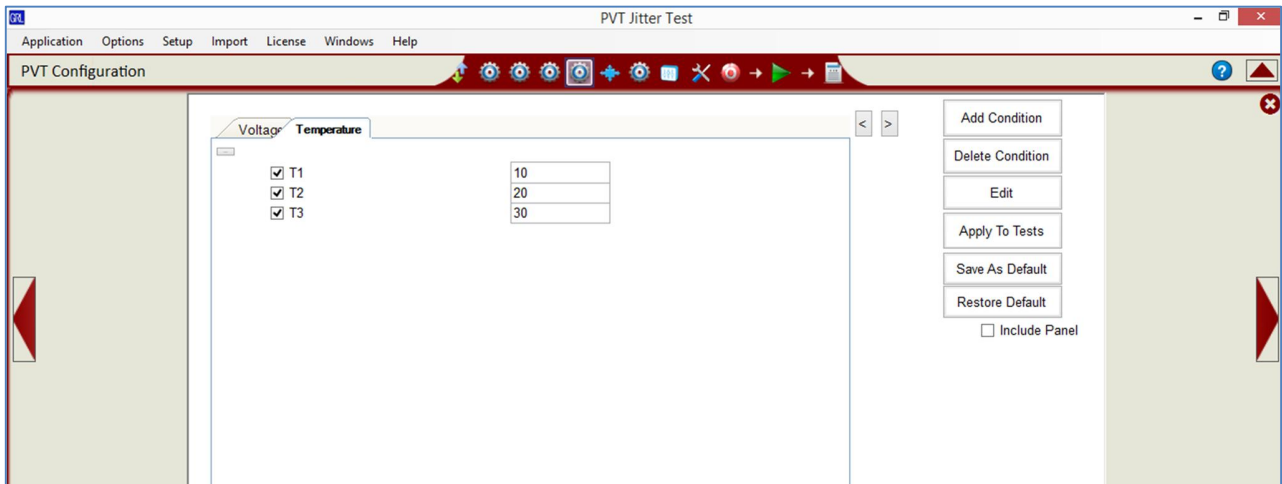


FIGURE 21. PVT CONFIGURATION EXAMPLE #3

4. To access the condition parameter in the code, use `FWAppObj.GetProperty("Condition")`. For example, `FWAppObj.GetProperty("Temperature")` should return T1, T2, or T3.
5. To access the value (or temperature tied to the condition), use `FWAppObj.GetProperty("EitherCondition_PVTValue")`. For example, `FWAppObj.GetProperty("T1_PVTValue")` returns "10" in this case.

4.8 Define Test Conditions

Select the  icon under the main menu to access the ‘Condition’ screen which can be useful when there is a need to repeat the same test without replicating/rewriting the code for each repetitive condition. For instance, in some circumstances, a DUT specific test plan might be required. Some DUT/standards might have different conditions to tests, and different conditions might have different specifications. In such a case, the user might want to define or organize a more DUT/standard set of specific conditions to handle the test. This can be done on the Conditions screen.

For example, a DisplayPort DUT requires tests to be carried out on 5.4 Gb/s, 2.7 Gb/s and 1.62 Gb/s data rates; and 400 mV, 600 mV, 800 mV output levels.

The Conditions screen allows the user to create a test plan/permutation table so that the user only needs to write code for a test once, and then the software can permute through all the conditions defined. The user can also define “condition-specific limit” for the test.

To add a condition:

1. Click on the **Add Condition** button.
2. Add a Condition Group and the condition to permute.

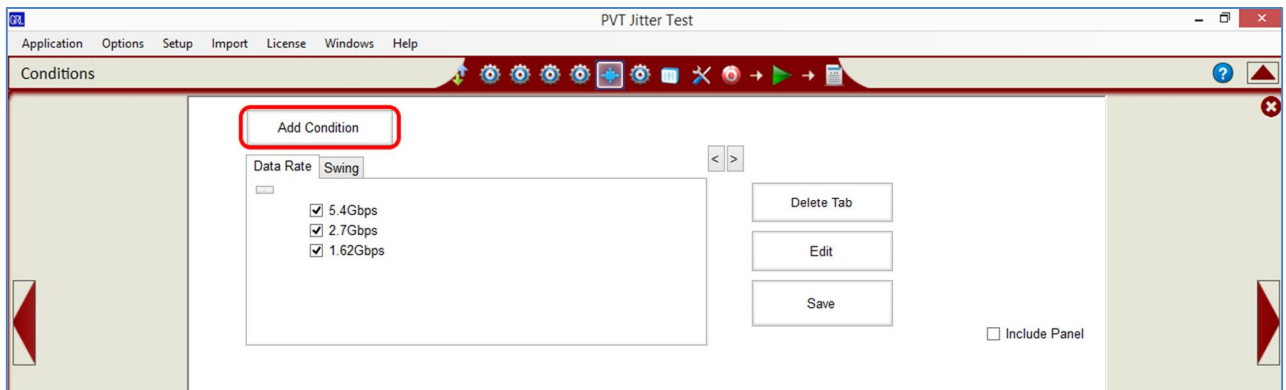


FIGURE 22. DEFINE TEST CONDITIONS EXAMPLE #1

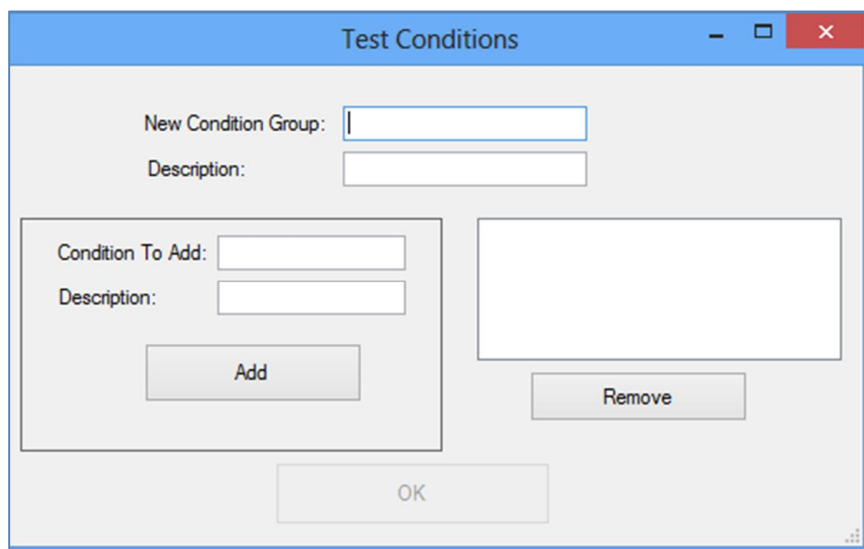


FIGURE 23. DEFINE TEST CONDITIONS EXAMPLE #2

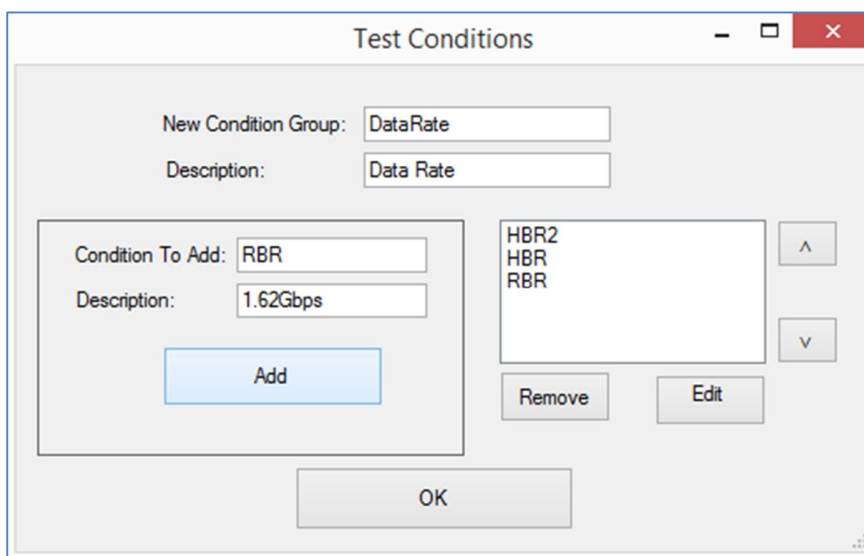


FIGURE 24. DEFINE TEST CONDITIONS EXAMPLE #3

4.9 Define Test Selections

Select the  icon under the main menu to access the 'Select Tests' screen which allows the user to create tests and test groups:

- Click on the **Add Group** button to add Test Groups. Always organize a group well for optimized test flow.
- Click on the **Add Test** button to add Tests.

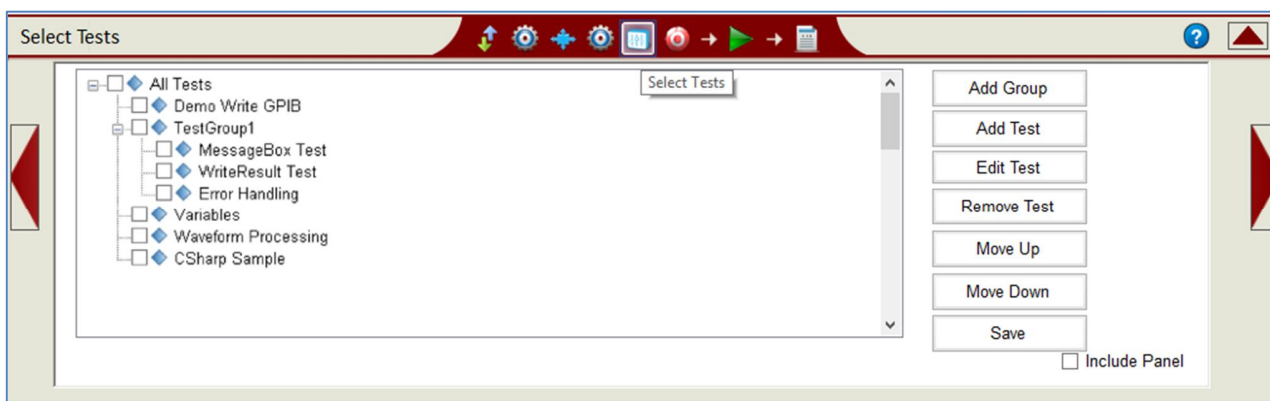


FIGURE 25. SELECT TESTS SCREEN

Once a test item is added, the user can click on the **Edit Test** button to define the test behavior.

4.9.1 Define Individual Tests

An example of the sequence for writing a new test is provided in Section 5.

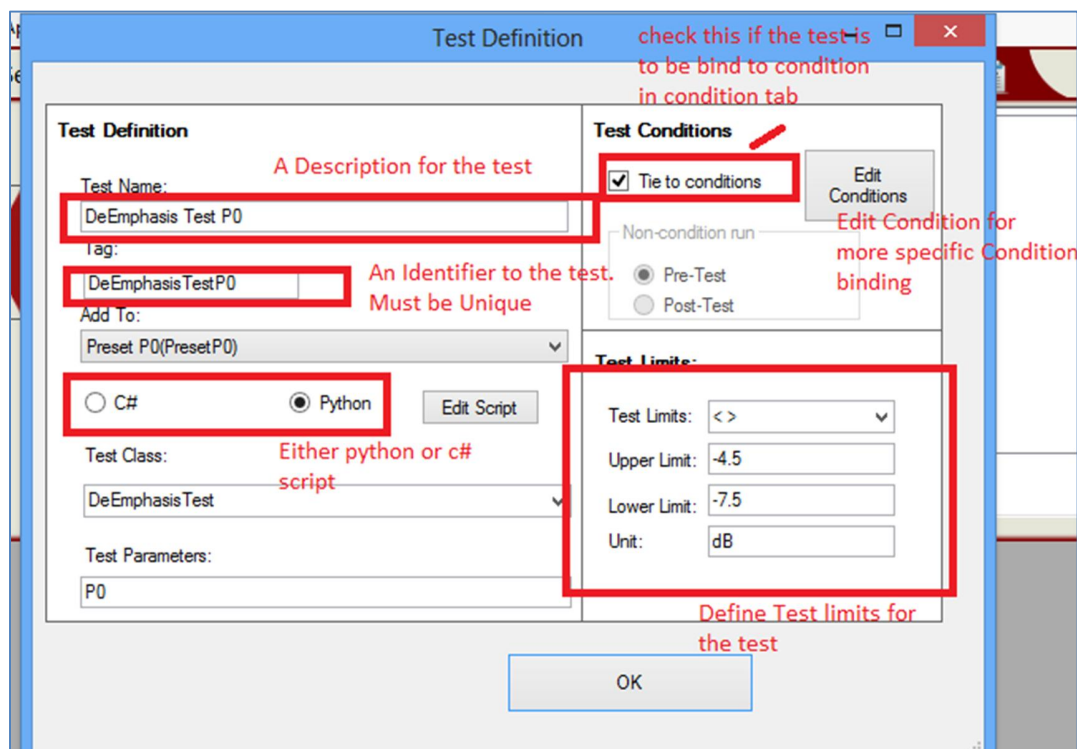


FIGURE 26. DEFINE TESTS EXAMPLE #1

To tie a test to conditions defined previously, select the **Tie to Conditions** checkbox, click on the **Edit Conditions** button, and choose the conditions the test needs to run.

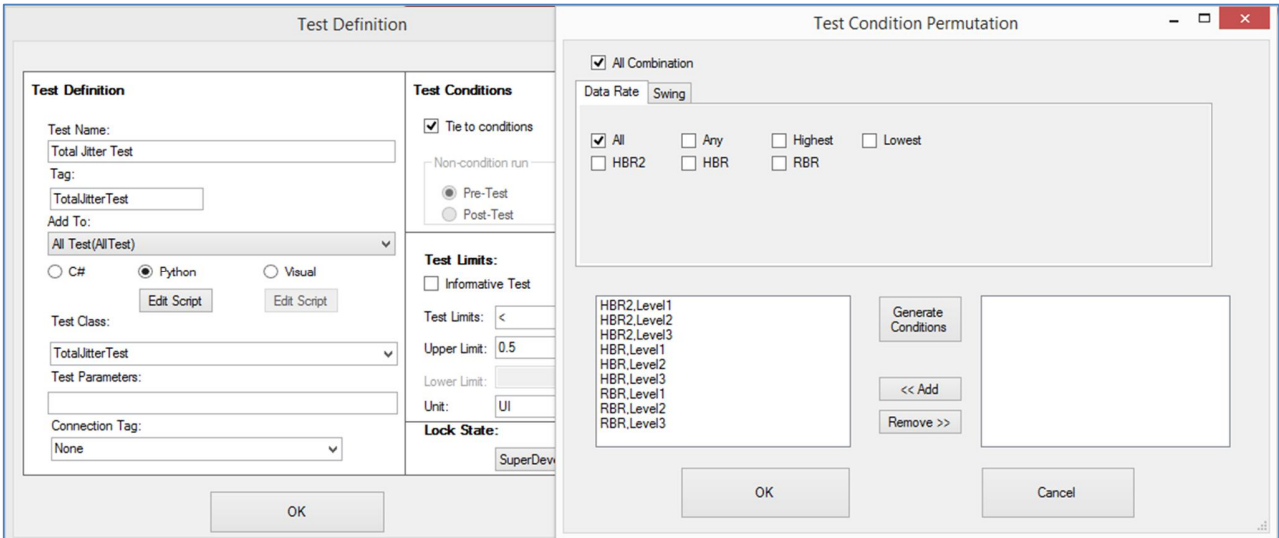


FIGURE 27. DEFINE TESTS EXAMPLE #2

When the Test Definition window opens, the user has the option to select the language used to write the test script, whether it is C#, Python or GRL Visual script.

Alternatively, there is also an option of 'Load Module'. This option allows the user to easily add tests by configuring the test flow through a user-friendly GUI with minimal knowledge in writing scripts.

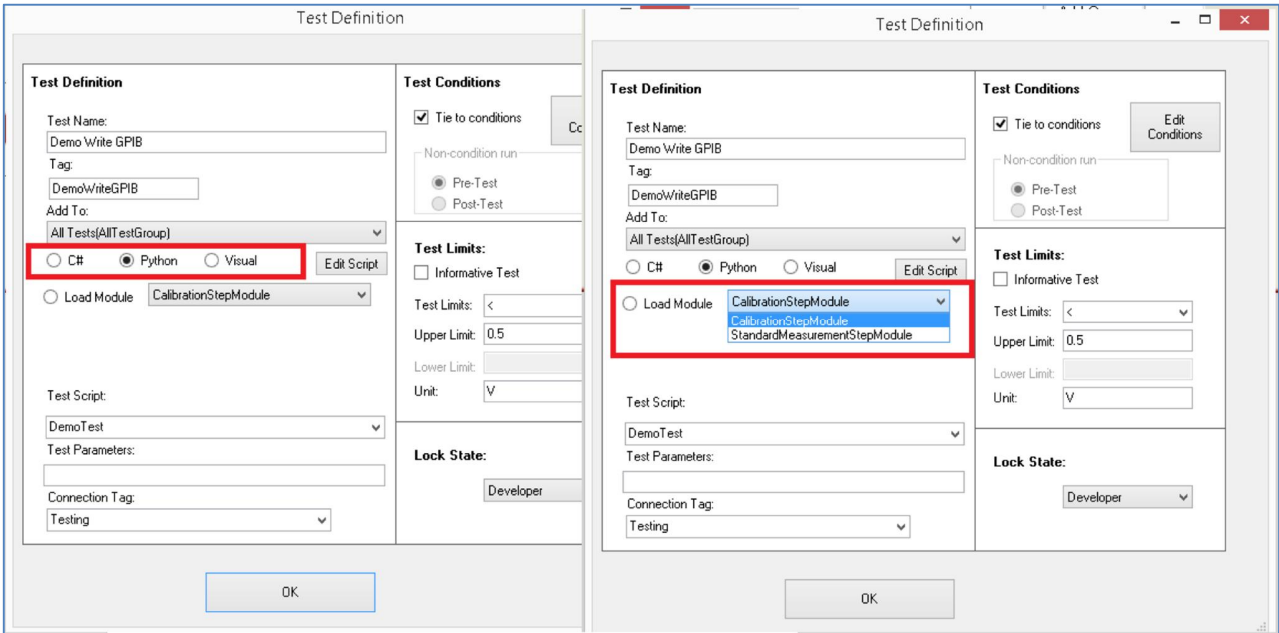


FIGURE 28. DEFINE TESTS EXAMPLE #3

4.9.2 Use an Existing Module

Predefined modules are available to ease the user's work to create common test functions. Predefined module availability will be based on the license options purchased by the user.

To use a predefined module, select **Load Module** and the module to be used. *Refer to Section 7, Using Predefined Module for more details.*

Click on the **Edit Script** button and a flowchart-like GUI will appear for the user to configure the test behavior. An example of "CalibrationStepModule" is as shown below.

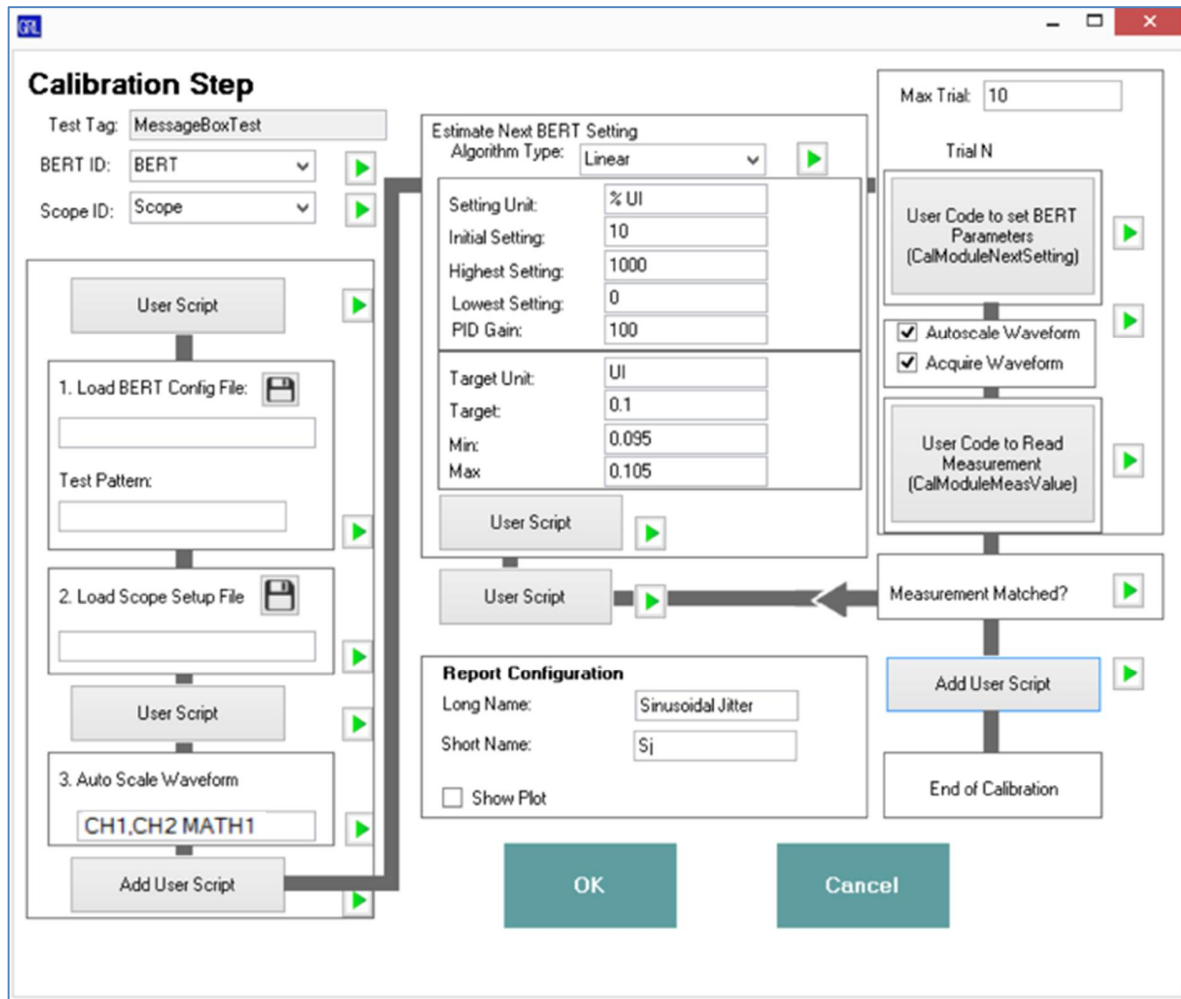


FIGURE 29. DEFINE TESTS EXAMPLE #4 – PREDEFINED MODULE

4.9.3 Define Test Groups

Figure 30 shows an example of how a test with the name "Rise Time" is being added to the Test Group "All Tests".

The user has selected **Add Group** using the button on the right of the Select Tests screen, and given the Test Group its name. Then the user selected **Add Test** to open the first Test Definition window.

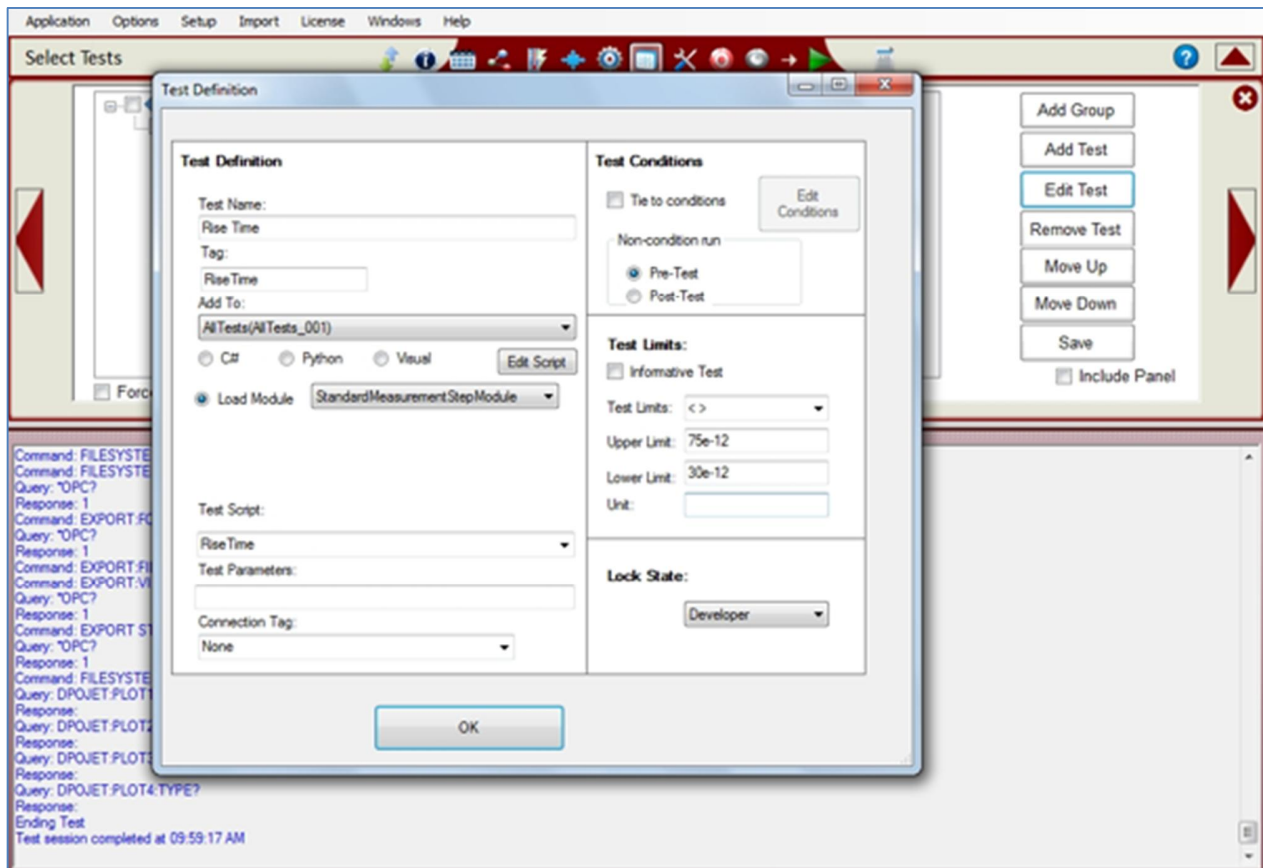


FIGURE 30. DEFINE TEST GROUP EXAMPLE

4.10 Define Setup Configurations

The user interface may be enhanced by including additional test parameter settings or a custom control.

On the 'Configurations' page, the developer can add custom GUI control components like label, textbox and combobox.

The control created can be accessed by `FWAppObj.GetProperty("ControlTag")`.

- Switch to **Developer Mode** to add/remove GUI control.
- Switch to **User Mode** to test the control added.

For instance, in the example in Figure 33, the user has defined a control called "DUTType" with Options of "Gen1" and "Gen2".

To access this control from script, one can use `FWAppObj.GetProperty("DUTType")`.

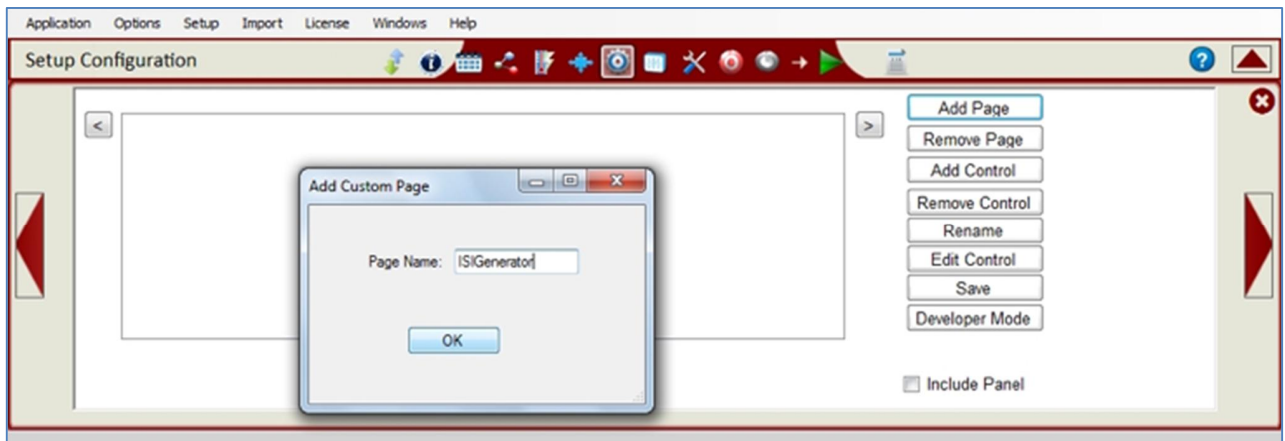


FIGURE 31. DEFINE SETUP CONFIGURATION– ADD CUSTOM PAGE

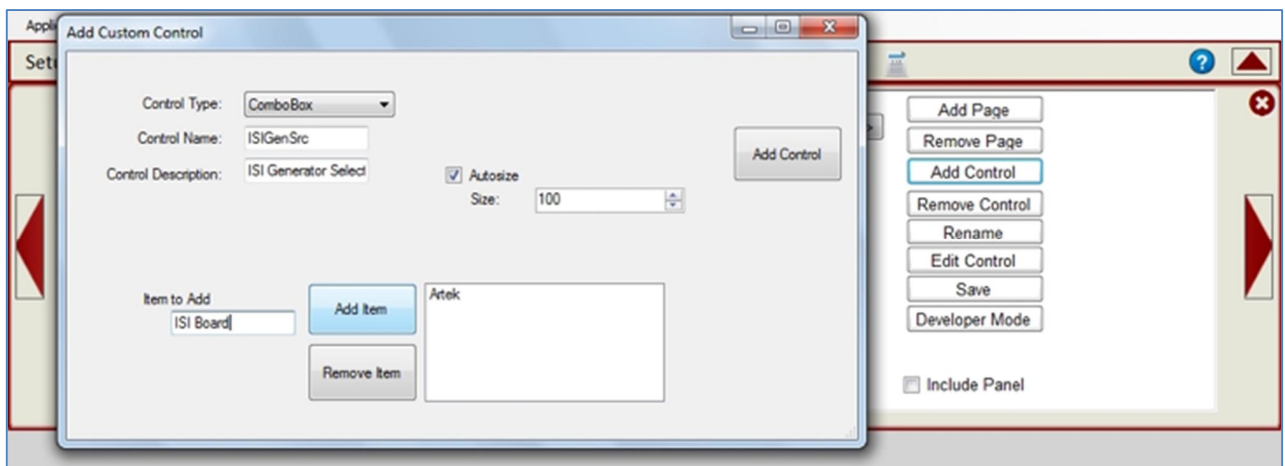


FIGURE 32. DEFINE SETUP CONFIGURATION– ADD CUSTOM CONTROL #1

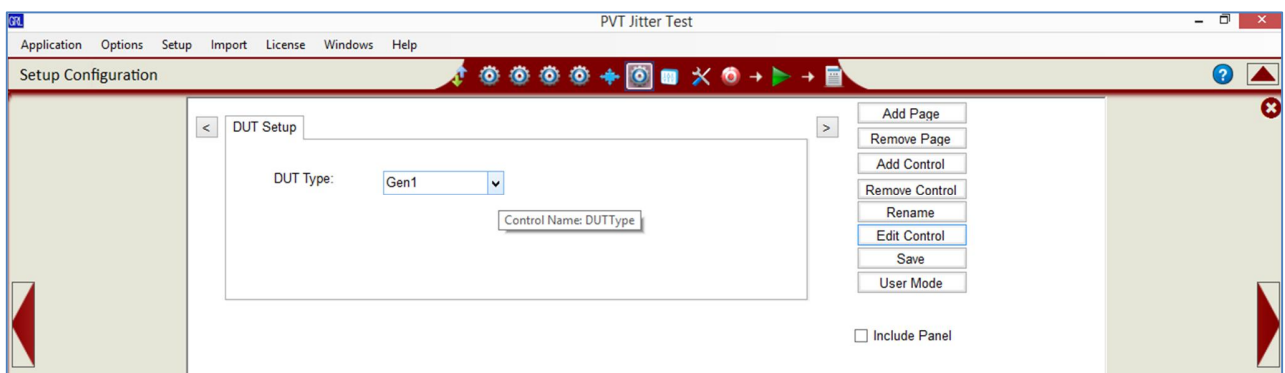


FIGURE 33. DEFINE SETUP CONFIGURATION– ADD CUSTOM CONTROL #2

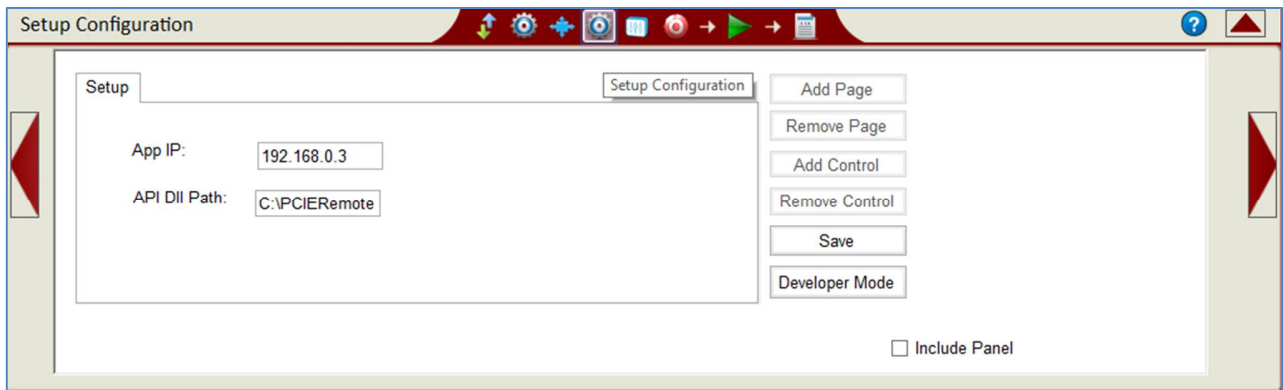


FIGURE 34. DEFINE SETUP CONFIGURATION– ADD SETUP

The Test Configurations tab allows the user to add configuration variables, which can later be accessed in Tree format. A ‘config’ can be a text box or a combo box.

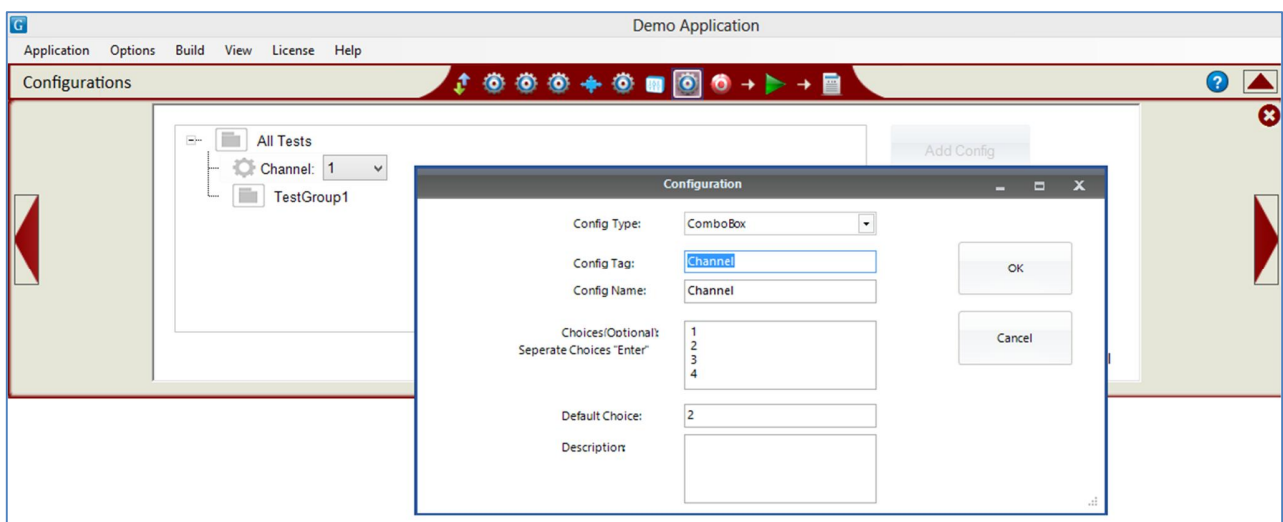


FIGURE 35. DEFINE SETUP CONFIGURATION– ADD VARIABLES IN TREE FORMAT

4.11 Define Compliance Limits

Select the  icon under the main menu to access the ‘Compliance Limits’ screen which allows the user to customize compliance limits for each test condition. This option is only available for those tests that are defined as “Tied to condition”.

To override a limit, click on the **Add Conditional Limits** button and then select the condition to be overridden.

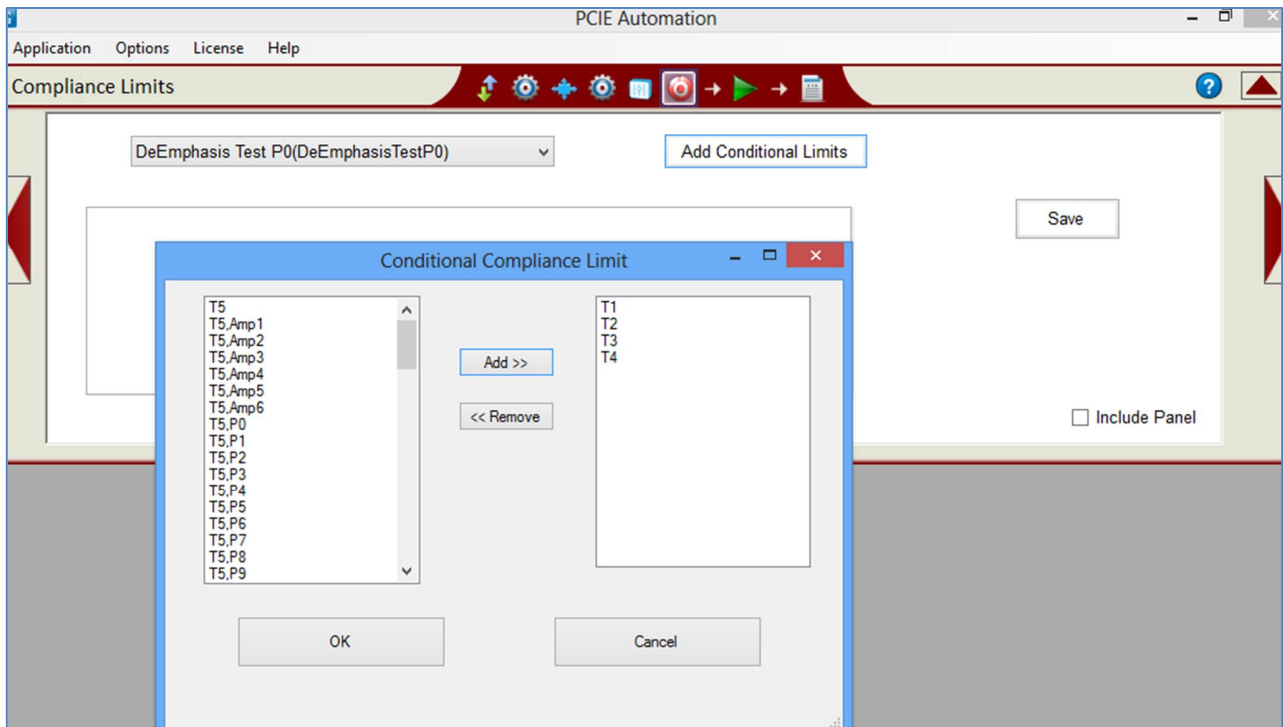


FIGURE 36. DEFINE COMPLIANCE LIMITS– ADD CONDITIONAL LIMITS EXAMPLE

To change the limit, select the condition and then edit the limits.

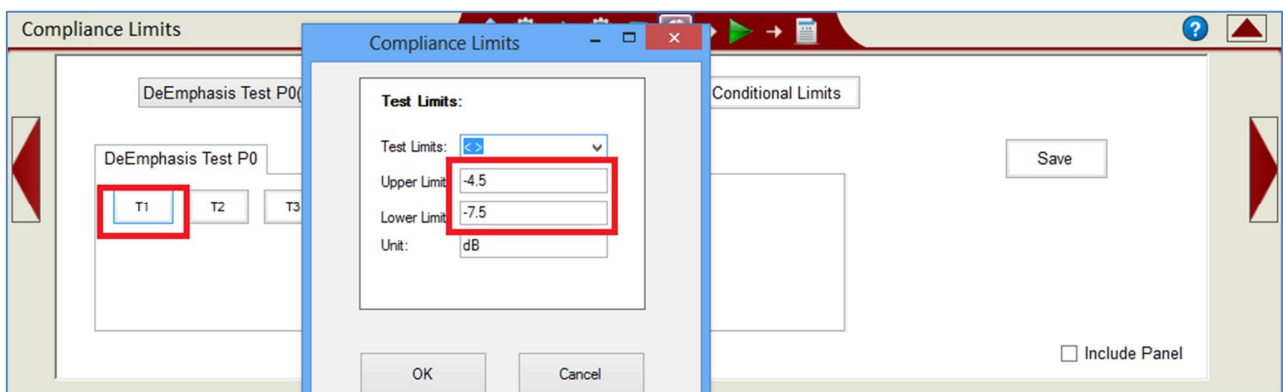


FIGURE 37. DEFINE COMPLIANCE LIMITS– CHANGE CONDITIONAL LIMITS EXAMPLE

4.12 Run Tests

Select the  icon under the main menu to access the 'Run Tests' screen which initiates the test run. Select the following options to run tests:

- **Skip Test if Result Exists** – If results from previous tests exist, the software will *skip* those tests. This allows the user to resume a test from previous test.
- **Replace if Result Exists** – If results from previous tests exist, the software will *replace* those tests with new results.

- **Delete Existing Results** – Delete all results before a test run begins, so that the user gets a fresh report.
- **Run Tests** – Run all the tests in a single trial.
- **Run Tests with PVT** – Run all the tests with all the conditions defined in the PVT Configuration screen.

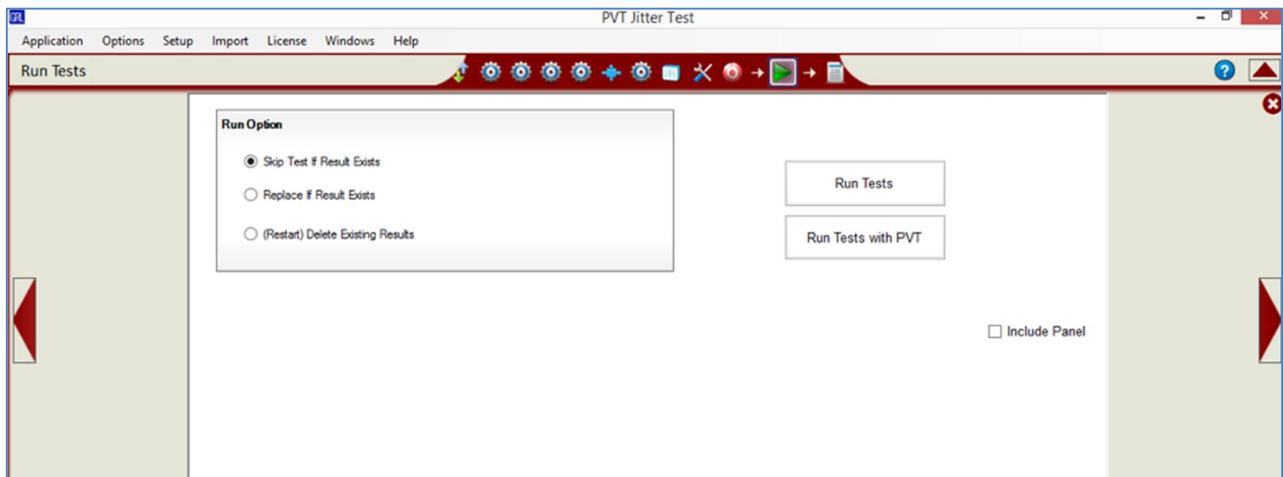


FIGURE 38. RUN TESTS SCREEN

4.13 Define Test Results and Reports

Select the  icon under the main menu to access the 'Report' screen which displays all the results from all the test runs.

If some of the results are not desired, they can be individually deleted by selecting the **Delete** button.

For detailed test report, click on the **Generate report** button to generate a PDF report. To have the calibration data plotted in the report, select the **Plot Calibration Data** checkbox.

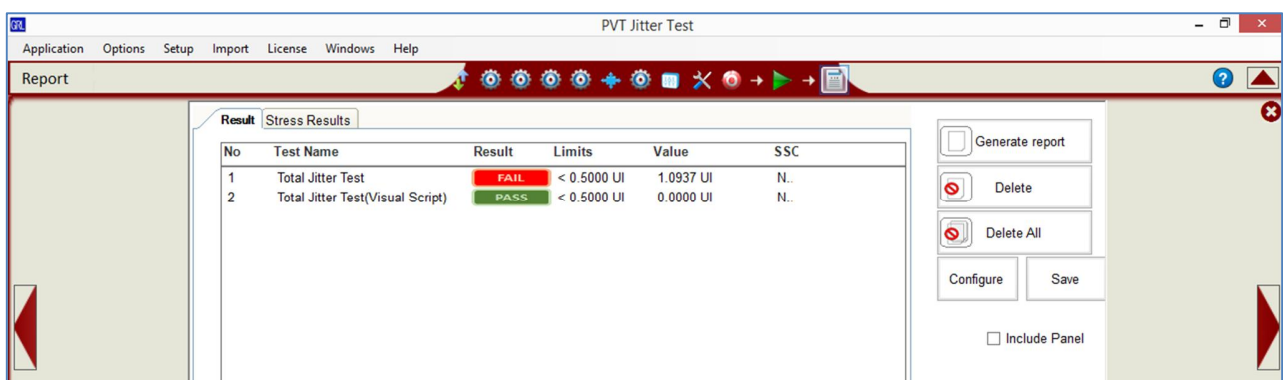


FIGURE 39. GENERATE REPORT SCREEN #1

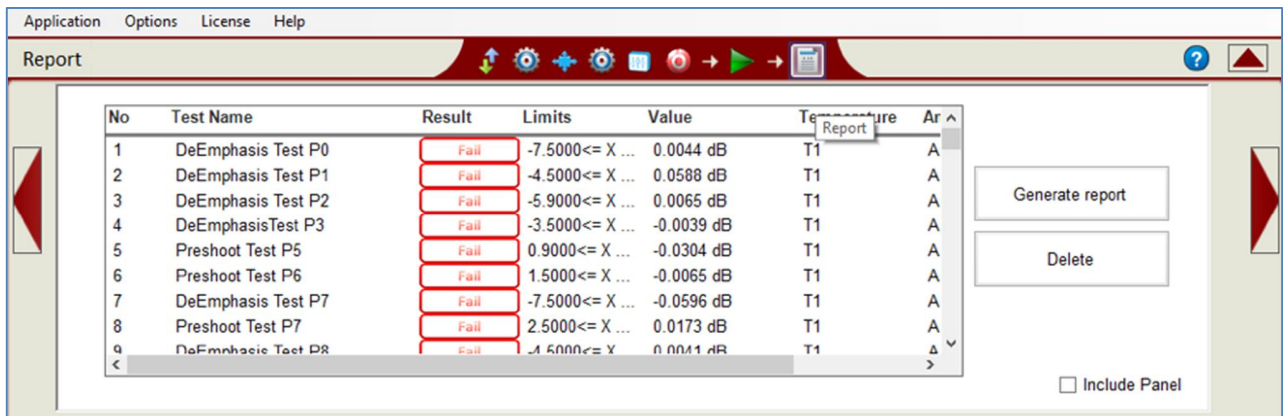


FIGURE 40. GENERATE REPORT SCREEN #2

4.13.1 Test Session Information

This portion displays the information previously entered on the **Session Info** page.

DUT Information	
DUT Manufacturer	: Company A
DUT Model Number	: Model B
DUT Serial Number	: SN01234567
Test Information	
Test Lab	: Lab A
Test Operator	: Operator 1
Test Date	: 12/01/2016
Software Version	
Software Revision	: 0.000.000.0001
Tek BERTScope FW	: 11.01
DPOJET Version	: 7.1.2.38
Tek Scope FW	: 7.6.0

FIGURE 41. TEST SESSION INFORMATION EXAMPLE

4.13.2 Test Summary Table

This table provides an overall view of all the tests performed along with their conditions and results.

No	TestName	Limits	Value	Results	Preset	SSC	SJLF
1	Launch Amplitude Calibration	True/False	True	Pass			
2	De-emphasis Calibration	True/False	True	Pass			
3	Pre-shoot Calibration	True/False	True	Pass			
4	Buj Calibration	True/False	True	Pass	N/A	SSC_ON	N/A
5	Buj Calibration	True/False	True	Pass	N/A	SSC_OFF	N/A
6	Rj Calibration	True/False	True	Pass	N/A	SSC_ON	N/A
7	Rj Calibration	True/False	True	Pass	N/A	SSC_OFF	N/A
8	Sj Calibration	True/False	True	Pass	N/A	SSC_ON	SJ1
9	Sj Calibration	True/False	True	Pass	N/A	SSC_ON	SJ2
10	Sj Calibration	True/False	True	Pass	N/A	SSC_ON	SJ3
11	Sj Calibration	True/False	True	Pass	N/A	SSC_ON	Custom_SJ1
12	Sj Calibration	True/False	True	Pass	N/A	SSC_ON	Custom_SJ2
13	Sj Calibration	True/False	True	Pass	N/A	SSC_ON	Custom_SJ3
14	Sj Calibration	True/False	True	Pass	N/A	SSC_OFF	SJ1
15	Sj Calibration	True/False	True	Pass	N/A	SSC_OFF	SJ2
16	Sj Calibration	True/False	True	Pass	N/A	SSC_OFF	SJ3
17	Sj Calibration	True/False	True	Pass	N/A	SSC_OFF	Custom_SJ1
18	Sj Calibration	True/False	True	Pass	N/A	SSC_OFF	Custom_SJ2
19	Sj Calibration	True/False	True	Pass	N/A	SSC_OFF	Custom_SJ3
20	Tj Calibration	True/False	True	Pass	N/A	SSC_ON	N/A
21	Tj Calibration	True/False	True	Pass	N/A	SSC_OFF	N/A
22	Insertion Loss	True/False	True	Pass			
23	Crosstalk Calibration	True/False	True	Pass			
24	RX Compliance Test	True/False	True	Pass	Preset_No_EQ	SSC_ON	SJ1
25	RX Compliance + Margin Test	True/False	True	Pass	Preset_No_EQ	SSC_ON	SJ1
26	RX Margin Search Test	True/False	True	Pass	Preset_No_EQ	SSC_ON	SJ1

FIGURE 42. TEST SUMMARY TABLE EXAMPLE

4.13.3 Test Results

This portion displays the results in detail along with supporting data points and screenshots for each test run.

22. Insertion Loss

Pass/Fail Stats	: Pass
Test Limits	: True/False
Result	: True
Cal Parameter	: InsertionLoss_
InsertionLossCal Settings	: 20.0000
Measured Value	: 23.2540 dB
Test completed time	: 05 April 2018 20:08:00 PM

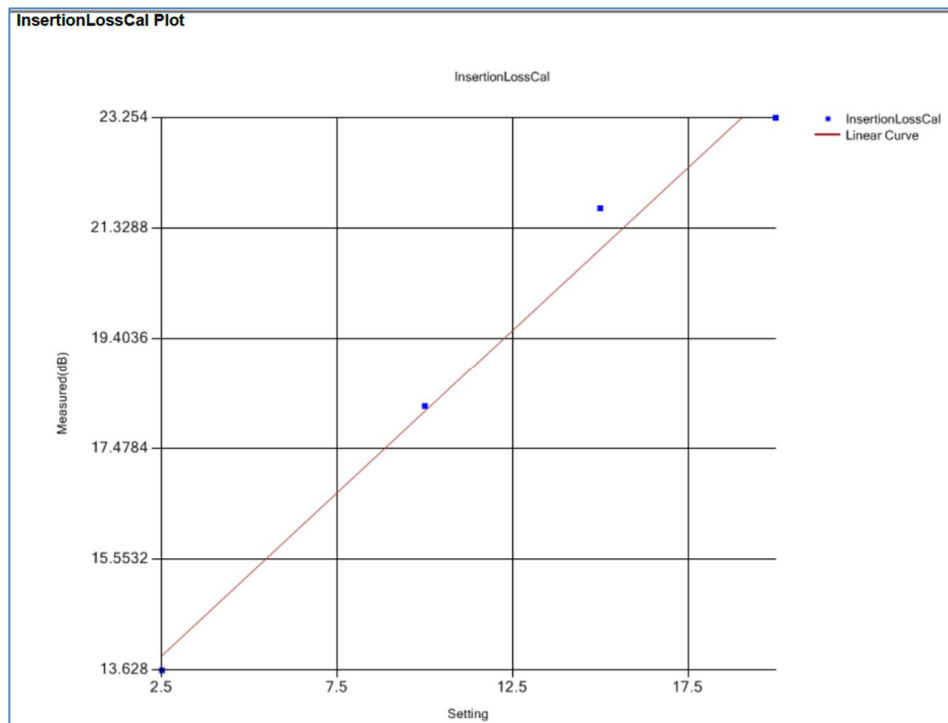


FIGURE 43. TEST RESULTS EXAMPLE

4.13.4 Delete Test Results

To individually delete any unwanted test results, select the corresponding result row and **Delete** button.

To entirely remove all existing test results, select the **Delete All** button.

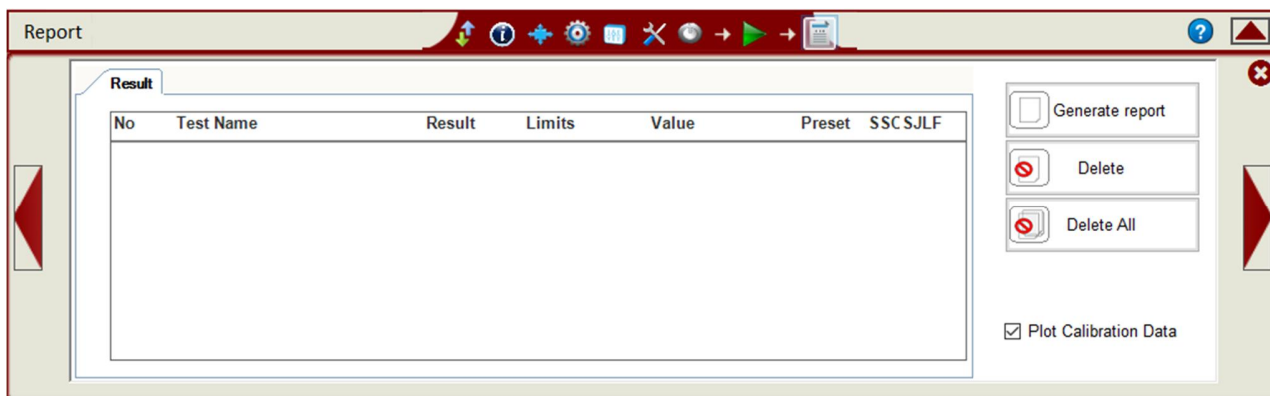


FIGURE 44. TEST REPORT DELETED

5 Example for Writing a New Test

Here shows an example of how to write a new test using the GRL KayaQ framework.

1. Add a Test

- Refer to Section 4.9, Define Test Selections to add a test item.

2. Edit a Test

- Once a test item is added, the user can click on the **Edit Test** button to define the test behavior.

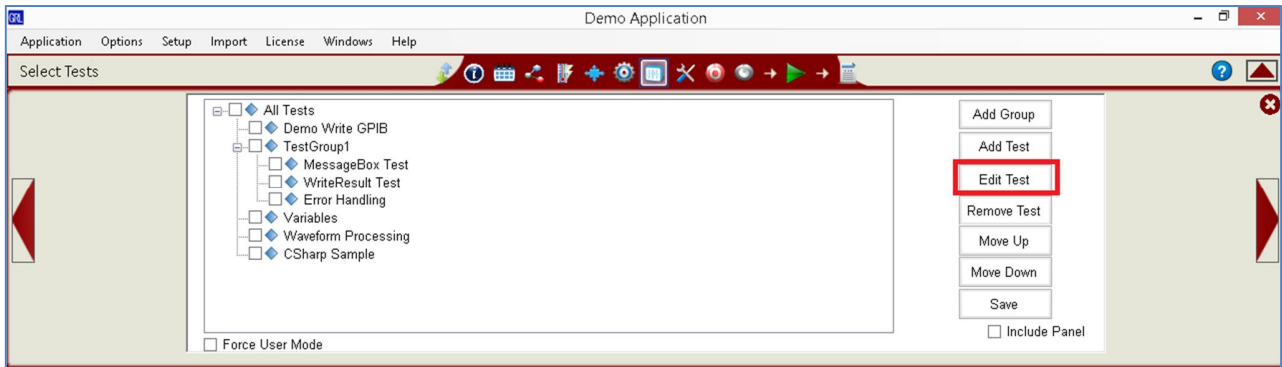


FIGURE 45. WRITING TEST EXAMPLE– EDIT TEST

3. Write/Define Test Behavior

- When the Test Definition window opens, the user has the option to select the language used to write the test script, whether it is C#, Python or GRL Visual script.
- Alternatively, there is also an option of “Load Module”. This option allows the user to easily add test by configuring the test flow through user friendly GUI with minimal knowledge in writing script.

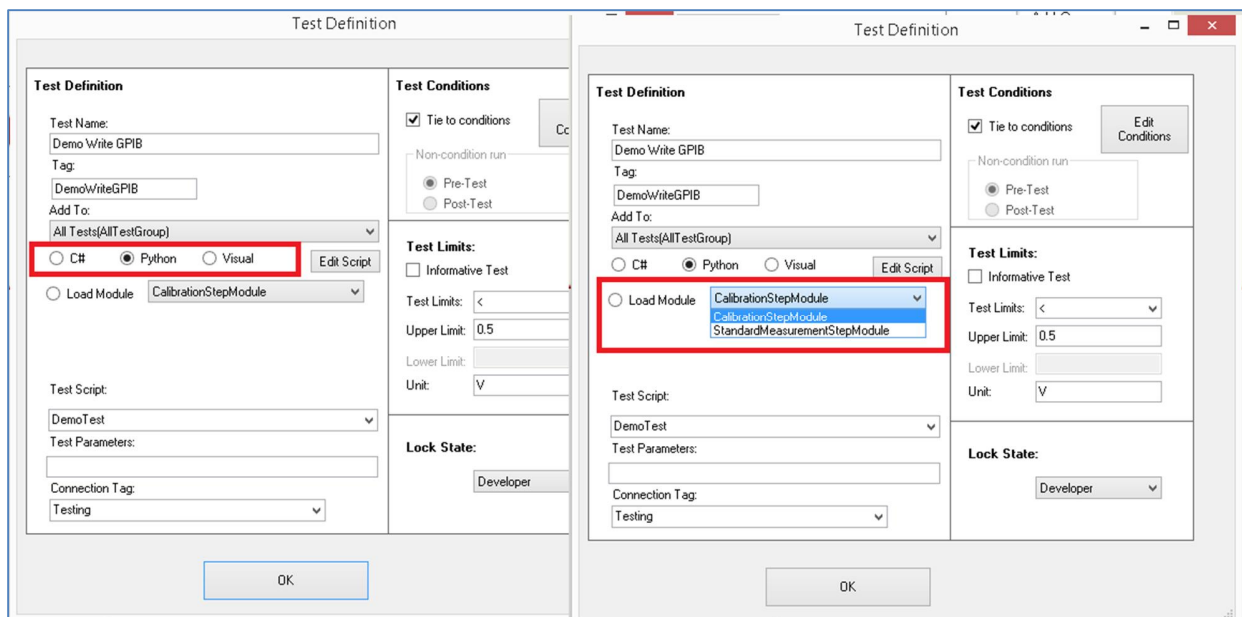


FIGURE 46. WRITING TEST EXAMPLE– WRITE/DEFINE TEST BEHAVIOR

a) Using C# Script

- A C# sample script is bundled with the installer and will be installed in the following location:

C:\GRL\GRL KayaQ Framework Test \Sample Projects\KayaQSampleProject.zip

- Unzip the folder and using Visual Studio, open the solution file.
- Below shows the files structure of the sample Project.

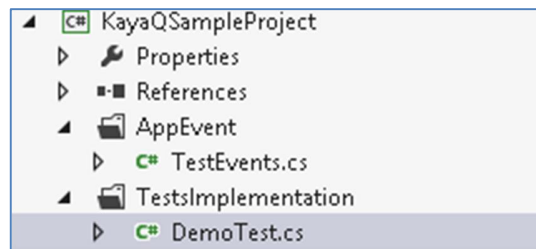


FIGURE 47. WRITING TEST EXAMPLE– USING C# SCRIPT #1

Now, relate the content of DemoTest.cs to the C# code.

- In the C# code, derive a class with the same name as test script name from the test definition window, in this case, “DemoTest”
- Override the “RunTest” function and all codes in the “RunTest” function will be called when the user runs the test.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using AppFrameWork.TestImplementation;
using AppFrameWork.ReportModuleObj;
using AppFrameWork;
using EquipmentLib;
using AppFrameWork.TestEquipmentObj;

namespace KayaQSampleProject
{
    public class DemoTest:TestWrapper
    {
```

The image shows a 'Test Definition' window with the following fields and settings:

- Test Name:** Demo\Write GPIB
- Tag:** Demo\WriteGPIB
- Add To:** All Tests(AllTestGroup)
- Script:** C# (highlighted with a red box)
- Load Module:** CalibrationStepModule
- Test Script:** DemoTest (highlighted with a red box)
- Test Conditions:** Tie to conditions (checked)
- Test Limits:** Informative Test (unchecked), Test Limits: <, Upper Limit: 0.5, Lower Limit: (empty), Unit: V
- Lock State:** Developer
- Buttons:** Edit Script, Edit Conditions, OK

FIGURE 48. WRITING TEST EXAMPLE– USING C# SCRIPT #2

b) Using Python Script

- After installation Python sample scripts can be found in: C:\GRL\GRL KayaQ Framework Test\Sample Projects\DemoApplication\scripts
- Here goes a brief description of each script:
 - DemoTest.py – simple code to teach the user to access Equipment added into the Equipment List and write a simple GPIB command.
 - TestMessageBox.py – simple code to teach the user to prompt different type of message boxes.
 - ErrorHandling.py – simple code to teach the user to pass the right value to the framework to handle events of error.
 - VariablesTest.py – simple code to teach the user to access GUI components of the Framework and set global variables.
- To define a test that runs Python scripts, select the **Python** option and define a test script name.
- When the **Edit Script** button is clicked, a Python script template will open.

Test Definition

Test Name:

Tag:

Add To:

☐ C#
 ☒ Python
 ☐ Visual

☐ Load Module

Test Script:

Test Parameters:

Connection Tag:

Test Conditions

☐ Tie to conditions

Non-condition run

☒ Pre-Test
 ☐ Post-Test

Test Limits:

☐ Informative Test

Test Limits:

Upper Limit:

Lower Limit:

Unit:

Lock State:

FIGURE 49. WRITING TEST EXAMPLE– USING PYTHON SCRIPT #1

```
import sys;
import FWTCPIPClient;
from FWAppObj import *
from ResultReport import *
from Equipment import *

FWAppObj.UpdateRunStats("Start Running Test\n")
print FWAppObj.GetCurrentTestName()
```

Add Code Here

FIGURE 50. WRITING TEST EXAMPLE– USING PYTHON SCRIPT #2

c) Using Visual Script

- To use a Visual script, select the **Visual** option and give a name to the script.
- Then click on the **Edit Script** button and a Visual Script Editor will open.
- Refer to Section 6, Using GRL Visual Scripting Tool for instructions on how to write a Visual script using the Editor.

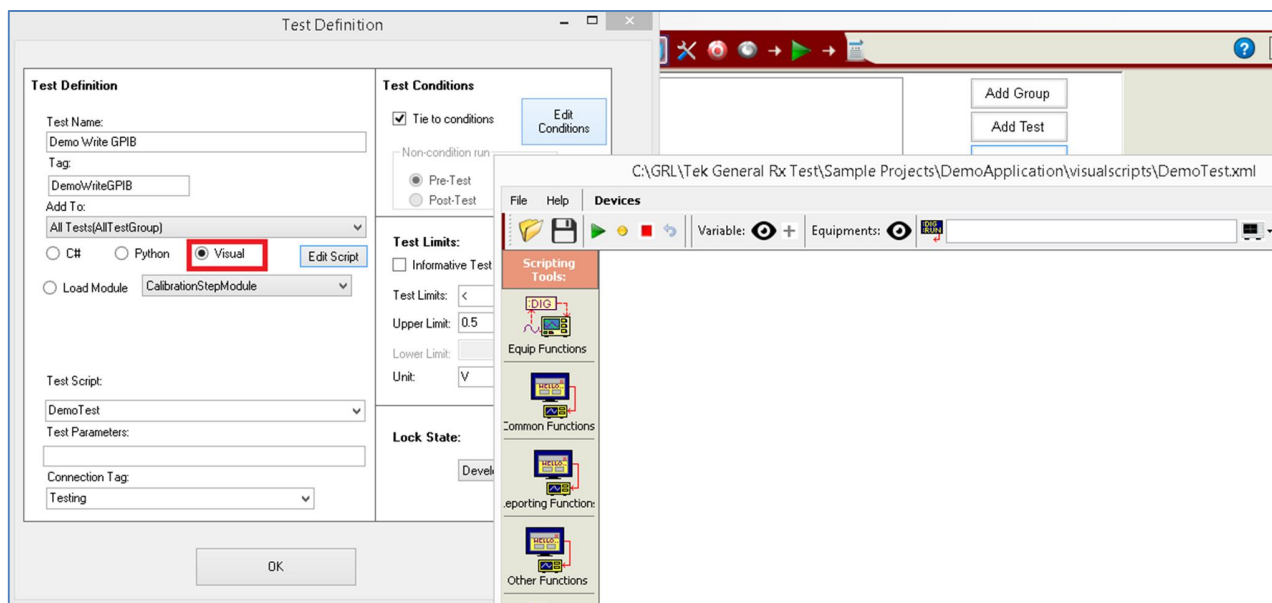


FIGURE 51. WRITING TEST EXAMPLE– USING VISUAL SCRIPT

d) Using Predefined Module

- Predefined modules are available to help the user to create a common test function. Predefined module availability will be based on the license option purchased by the user.
- To use a predefined module, select the **Load Module** option and select an existing module.
- Click on the **Edit Script** button and a flowchart-like GUI will appear for the user to configure the test behavior.
- Refer to Section 7, Using Predefined Module 4 for more details.
- Shown below is a UI screenshot of “CalibrationStepModule”.

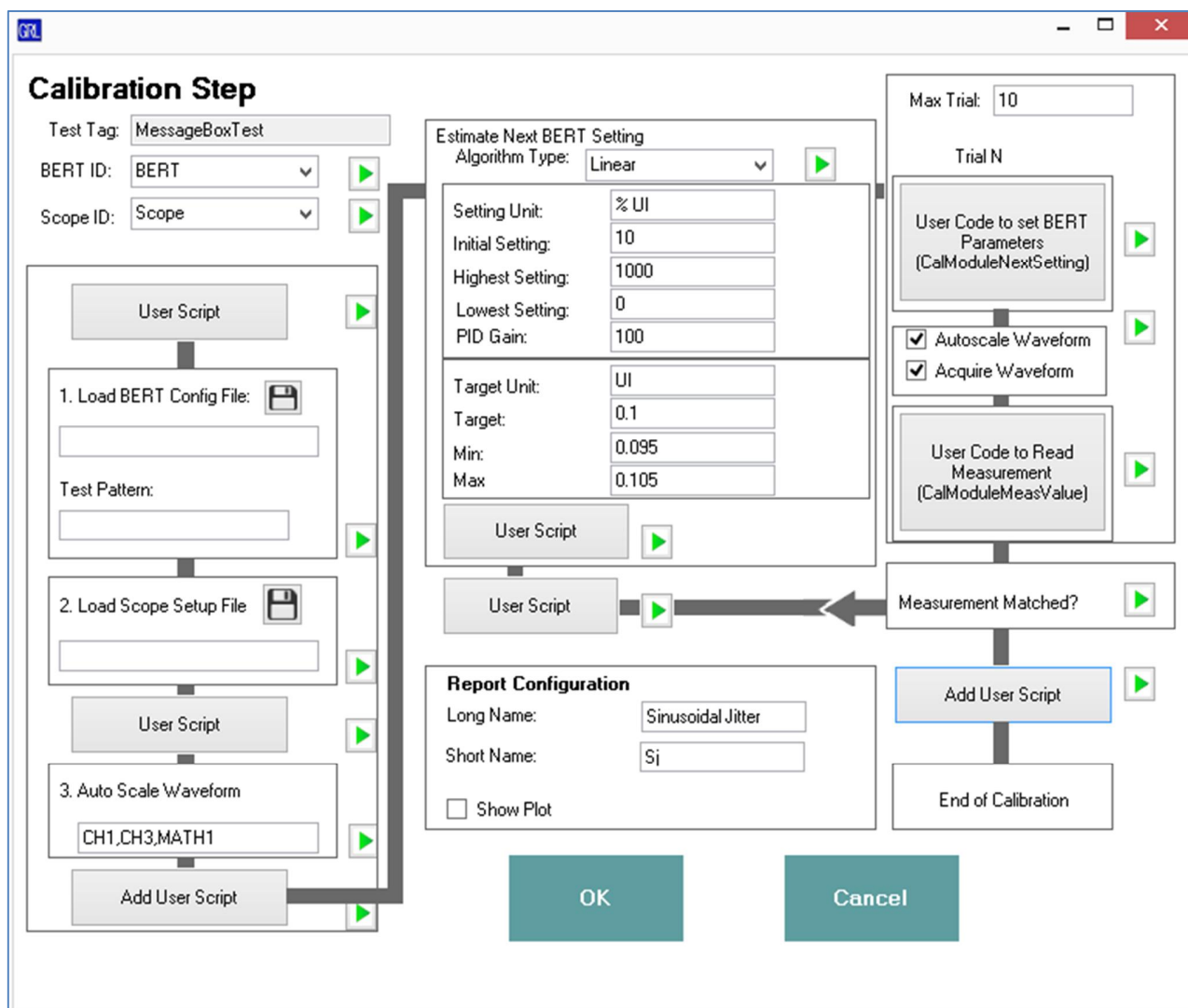


FIGURE 52. WRITING TEST EXAMPLE– USING PREDEFINED MODULE

6 Using GRL Visual Scripting Tool

While the GRL KayaQ framework utilizes third party tools e.g., Python and Visual Studio for C# scripts, the framework is also integrated with GRL's proprietary Visual scripting tool. The GRL Visual scripting tool is a user interactive based tool that allows developer to create scripts in a flowchart-like manner.

1. Launch the Visual script tool by:

- clicking on the  icon on the Equipment Setup screen and then clicking on  again, or
- clicking on the **Edit Script** button in the Test Definition window when the option to Edit script is available.

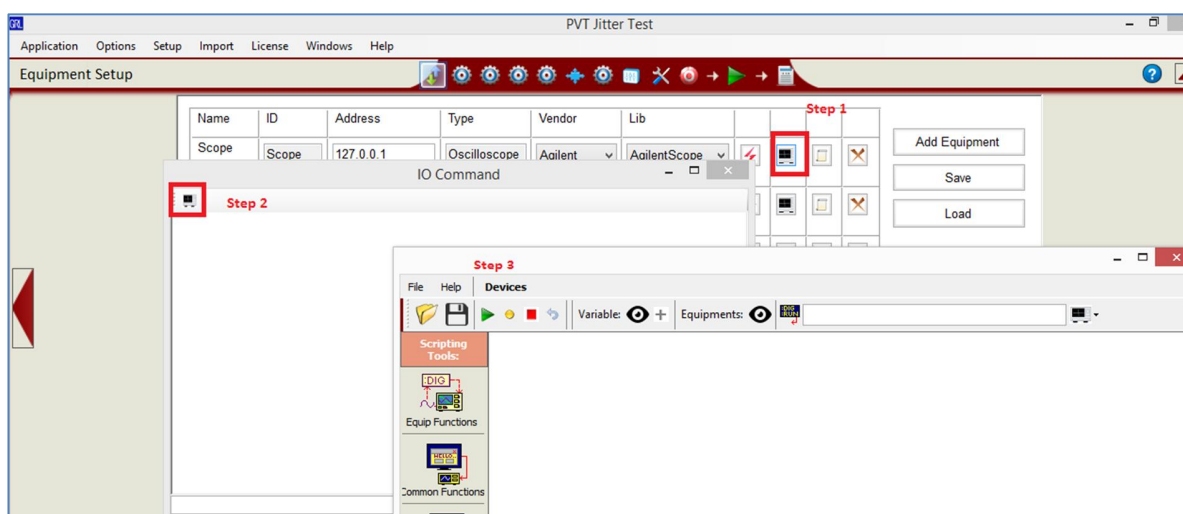


FIGURE 53. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– LAUNCH VISUAL SCRIPT TOOL #1

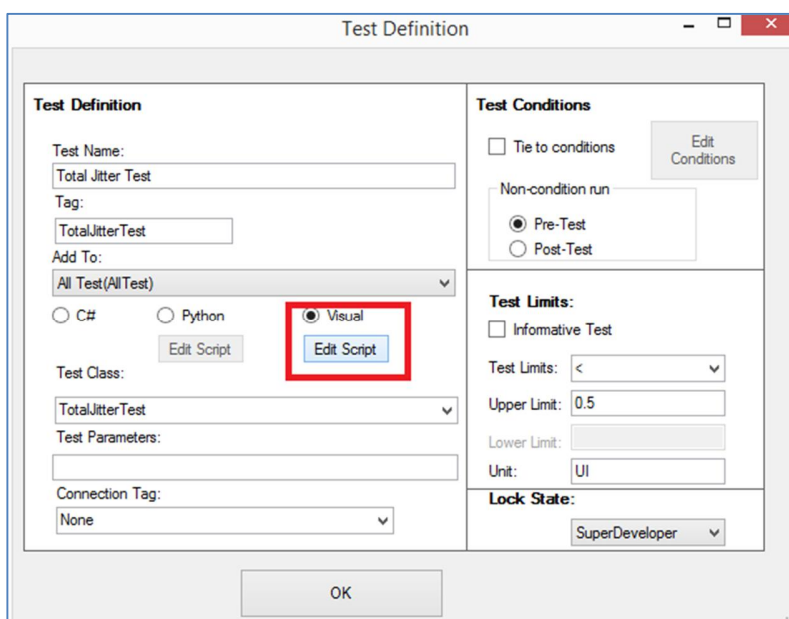


FIGURE 54. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– LAUNCH VISUAL SCRIPT TOOL #2

2. The following diagram describes the features of the GRL Visual scripting tool:

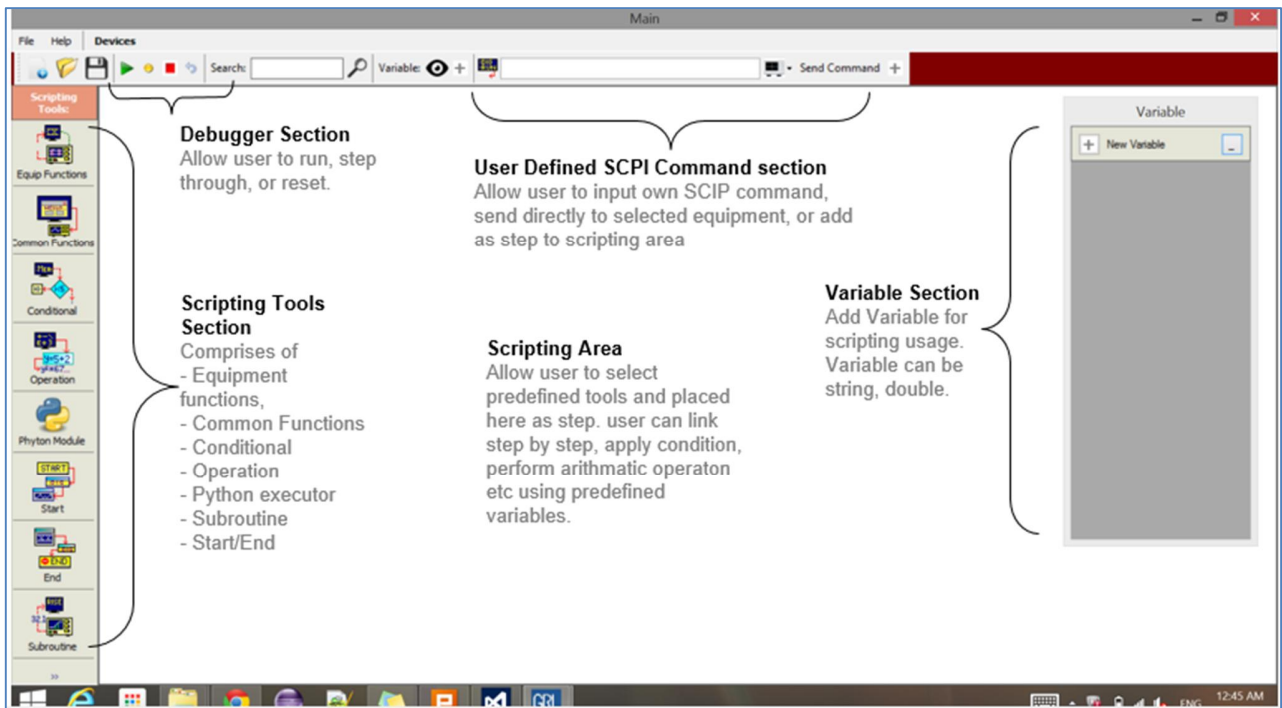


FIGURE 55. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– VISUAL SCRIPT TOOL FEATURES

3. The user can utilize local variables using the Visual scripting tool as shown below:

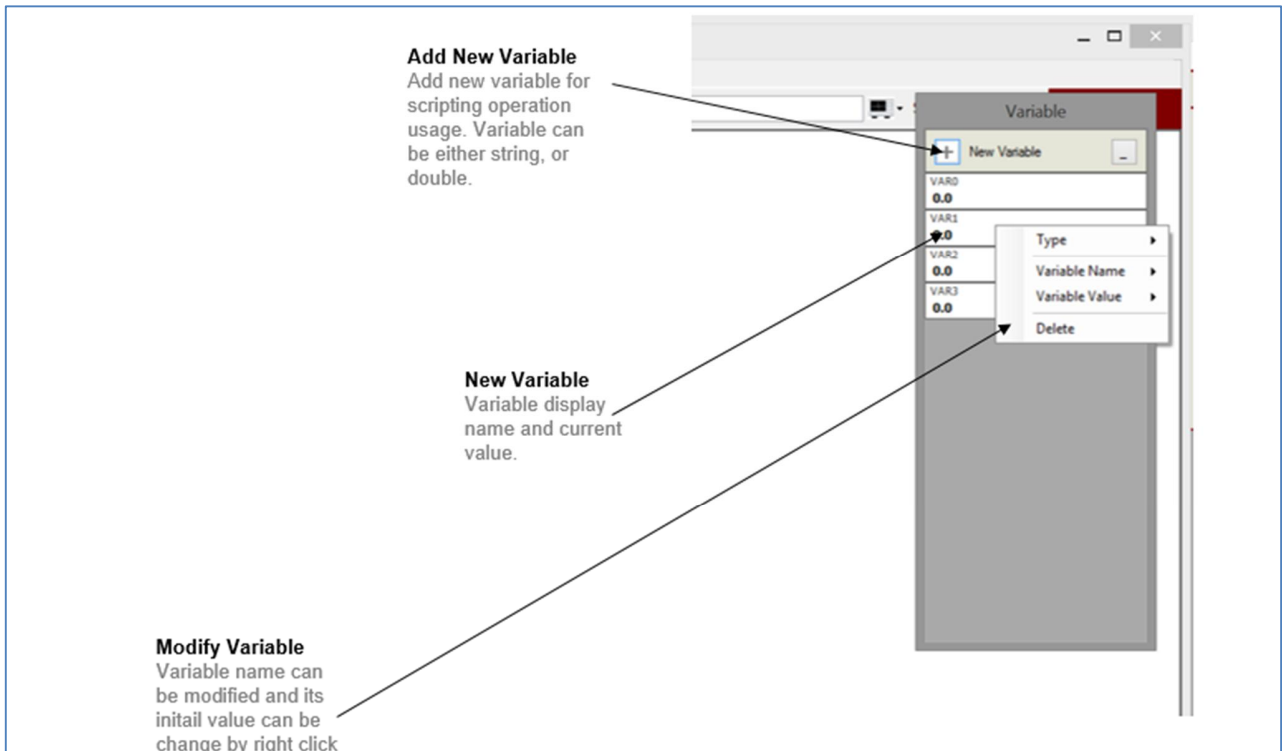


FIGURE 56. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– USING VARIABLES

4. To add a step node, the user can click on the buttons at the left panels to view available API's and select a node to add.

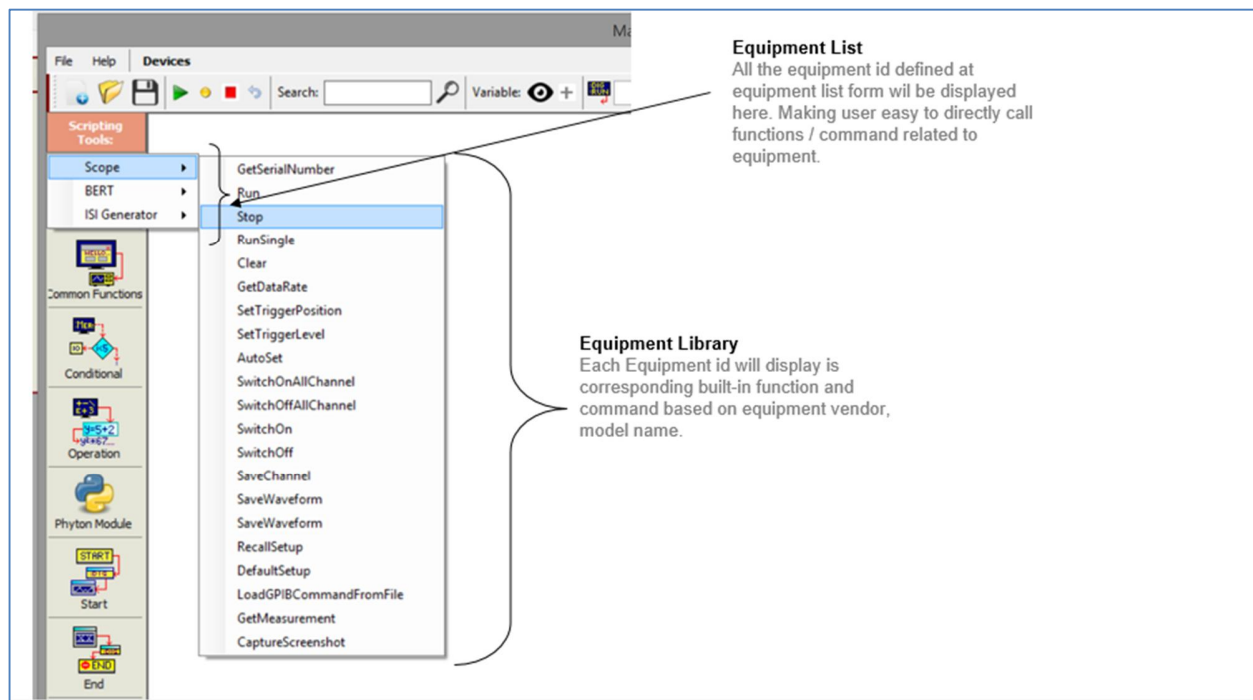


FIGURE 57. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– ADD NODE #1

5. Other operational steps include:
 - Start Node
 - End Node
 - If-Else Node
 - Node to call Python script
 - Common arithmetic node

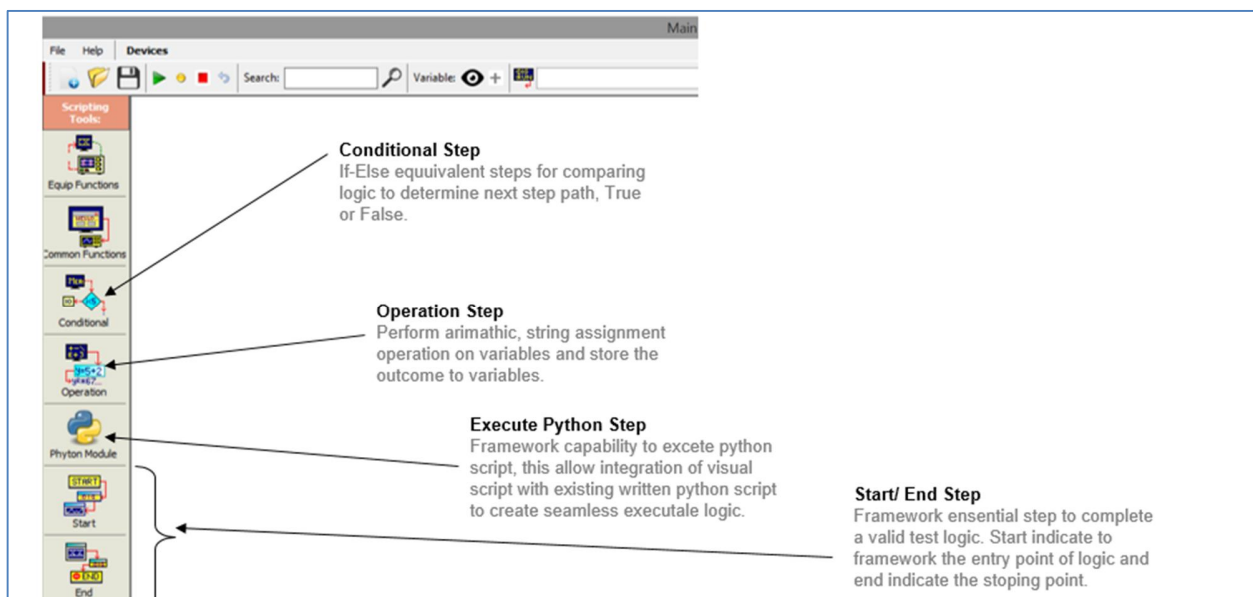


FIGURE 58. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– ADD NODE #2

6. The user can also perform debugging and stepping when using the GRL Visual script.

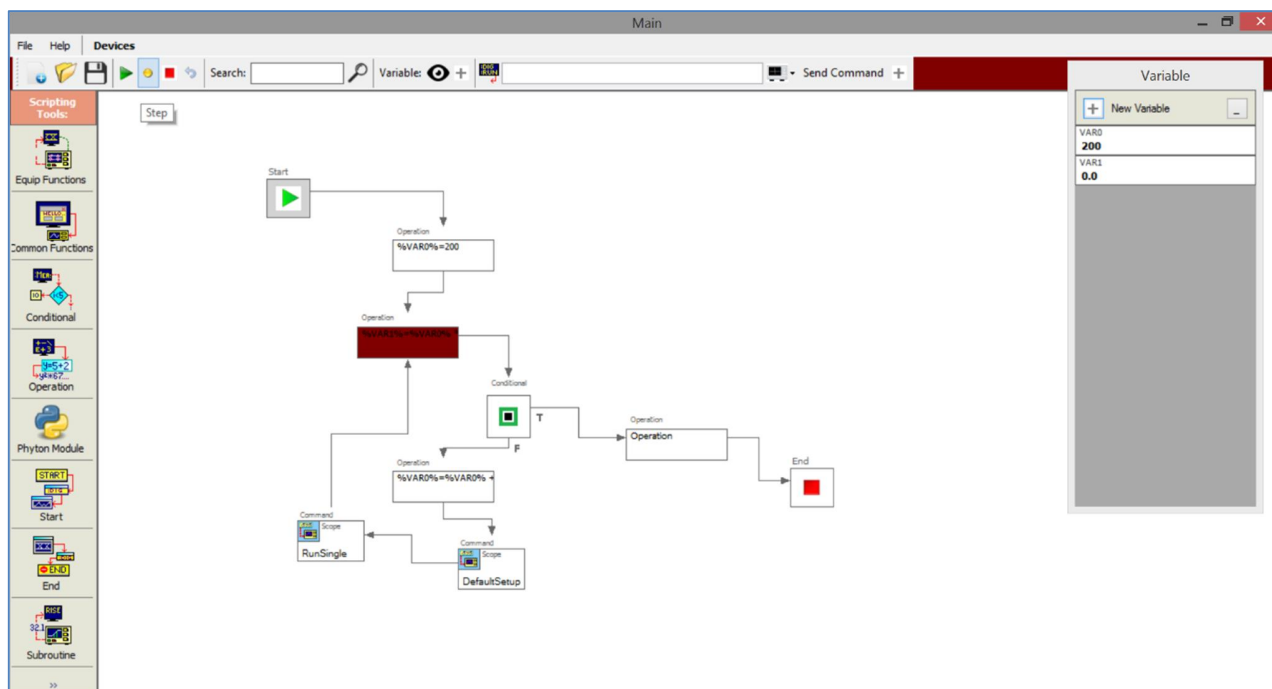


FIGURE 59. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– PERFORM DEBUGGING AND STEPPING

6.1 Programming Example

This section provides an example to write/send commands using the GRL Visual scripting tool.

1. The left panel of the GRL Visual scripting tool enables remote programming through the following three main function groups:
 - Common Functions– Provide access to the GRL KayaQ Framework user interface, tests items and status as well as other commonly used functions.
 - Reporting Functions– Provide access to reports of the GRL KayaQ application data.
 - Equipment Functions– Provide access to functions of the equipment that is added to the GRL KayaQ Framework.

These main function groups are indicated in Figure 60 below.

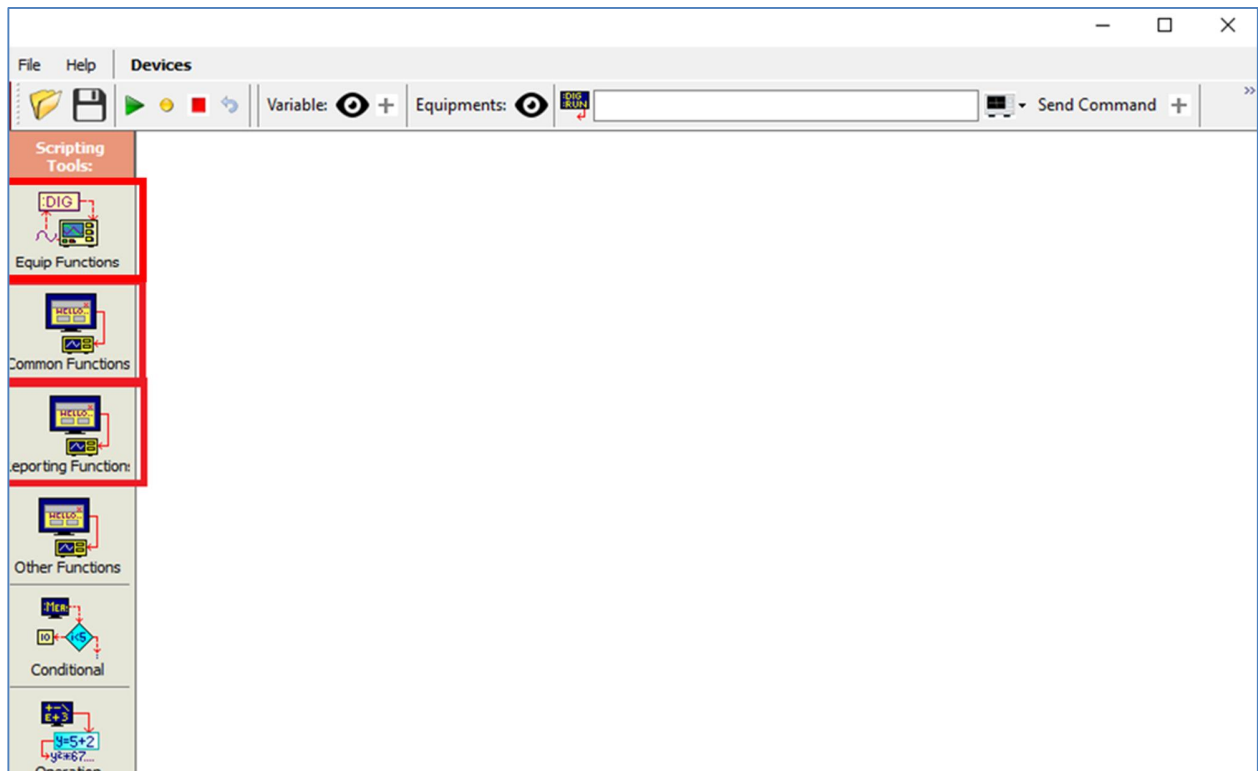


FIGURE 60. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– SELECT MAIN FUNCTION GROUPS FOR PROGRAMMING

2. Click on any of the main function groups to view available API's and select a node to add:

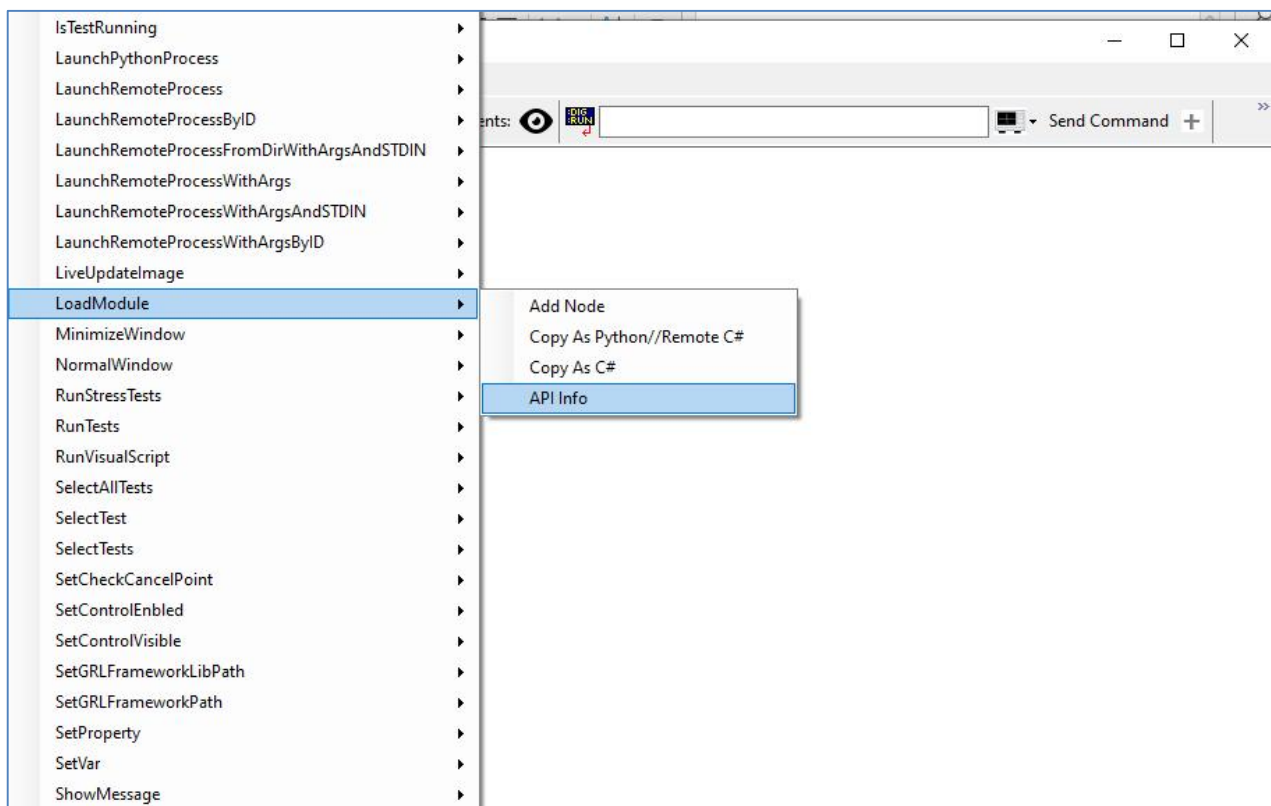


FIGURE 61. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– ACCESS MAIN FUNCTION GROUP API'S

3. To access detailed information for an API function, click on the **API Info** option:

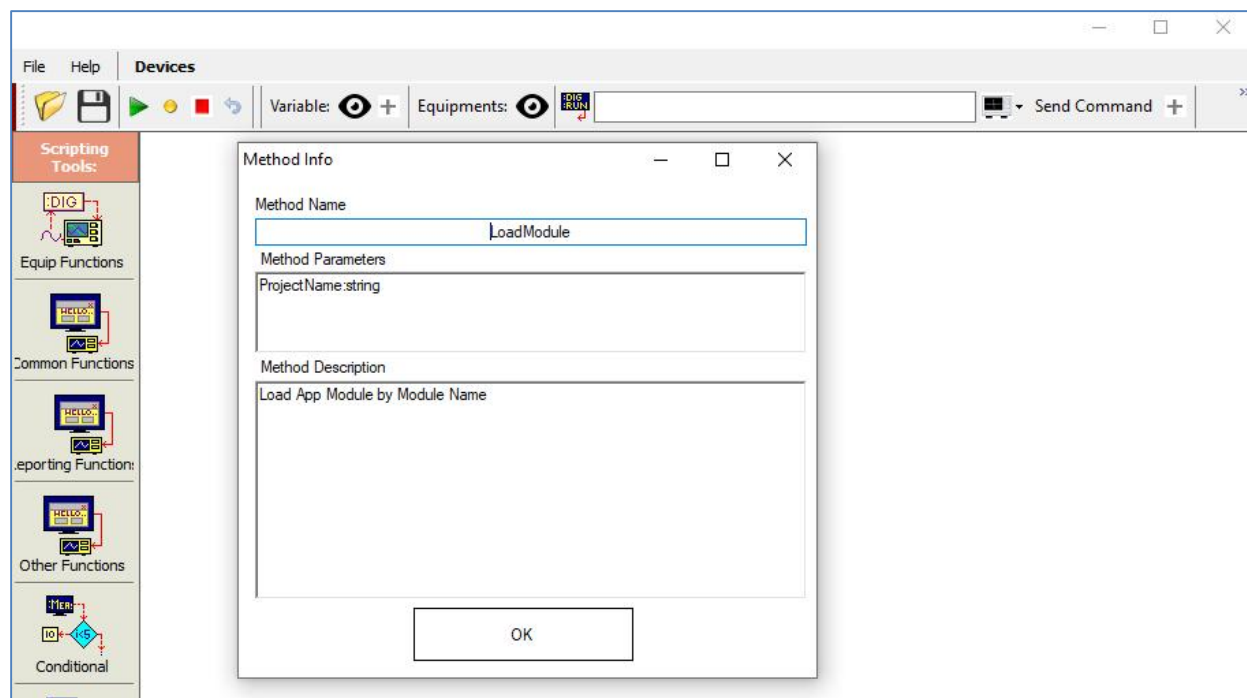


FIGURE 62. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– ACCESS API INFO

4. To automatically copy an existing command line to clipboard to be pasted directly on to a test script, click on the **Copy As Python/Remote C#** or **Copy As C#** option:

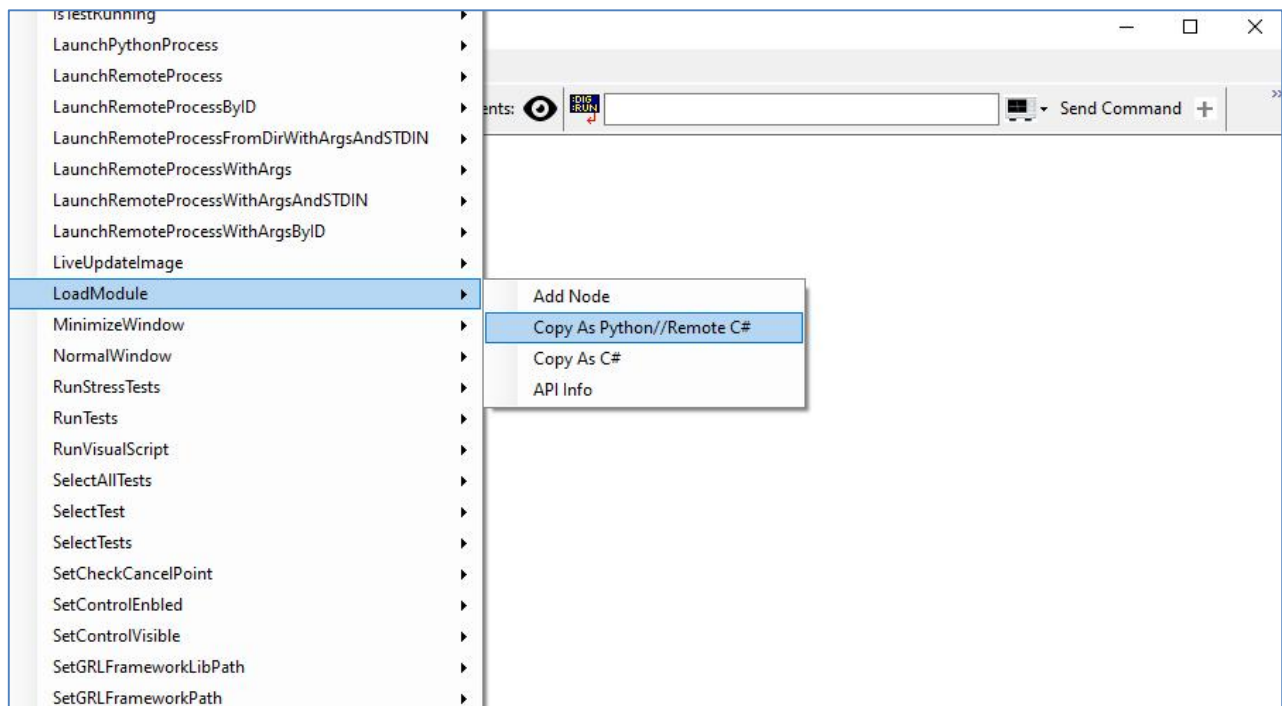


FIGURE 63. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– COPY COMMAND LINE TO CLIPBOARD

5. For example if a Python code is copied, paste the code on to a test script and change the parameters. The fraction code as shown below provides the sequence to load the GRL USB 3.1 Rx Module, select a test item and execute the test:

```
import sys;
from FWTCPIPClient import *;
import TCPIPCommand;
import json;
from FWAppObj import *
from ResultReport import *
from Equipment import *
import time;

FWTCPIPClient.Timeout=0;

FWTCPIPClient.IpAddress = '127.0.0.1'; #IP Address of PC where GRL Application is
installed
FWTCPIPClient.PortNo = 53355;

FWAppObj.SendCommand("LoadModule","Anritsu USB 3.1 Rx Test");
FWAppObj.SendCommand("SetProperty",["Gen2_10G",True]);
FWAppObj.SendCommand("SelectTests","DeEmphasisCalibration");
FWAppObj.SendCommand("RunTests", None);
```

The following explains each command line of the fraction code:

- a) **FWAppObj.SendCommand("LoadModule","Anritsu USB 3.1 Rx Test");**
- Loads the “Anritsu USB 3.1 Rx Test” application as follows:

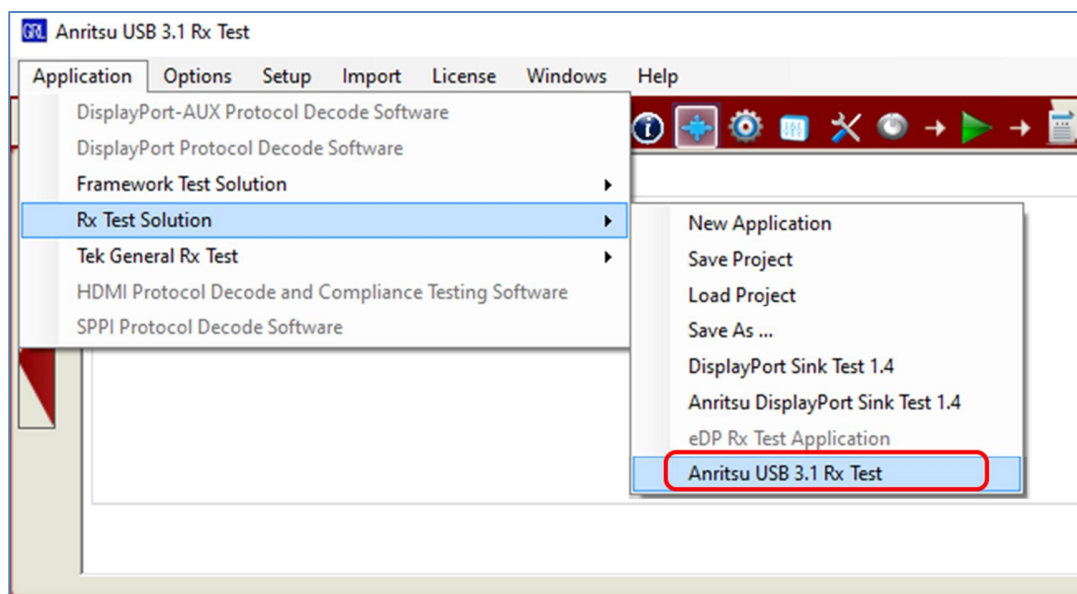


FIGURE 64. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– FRACTION CODE COMMAND #1

b) `FWAppObj.SendCommand("SetProperty", ["Gen2_10G", True]);`

- Sets the Checkbox property value of “Gen2_10G” to “True”.
- This command can be used to set any value of user interface control items (Text box, Checkbox, List box, etc.) of the GRL KayaQ Framework.
- Right-click on a user interface control item and click on “Show Hint” for the available value to set as follows:

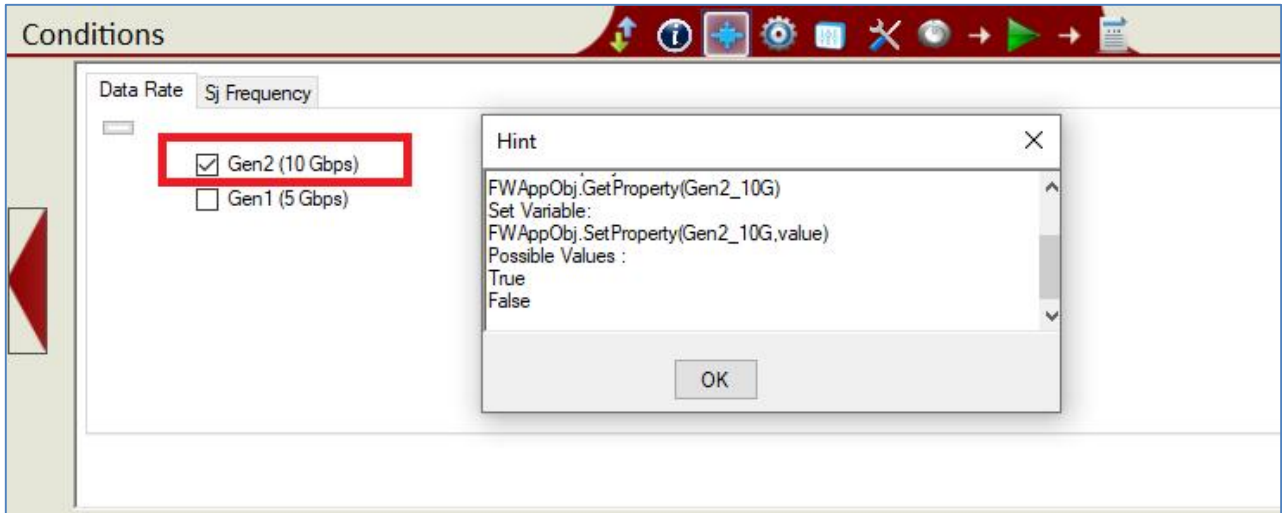


FIGURE 65. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– FRACTION CODE COMMAND #2

c) `FWAppObj.SendCommand("SelectTests", "DeEmphasisCalibration");`

- Selects the “Deemphasis Calibration” test item.
- Right-click on the test item to enable the test tag to be input as the function parameter as follows:

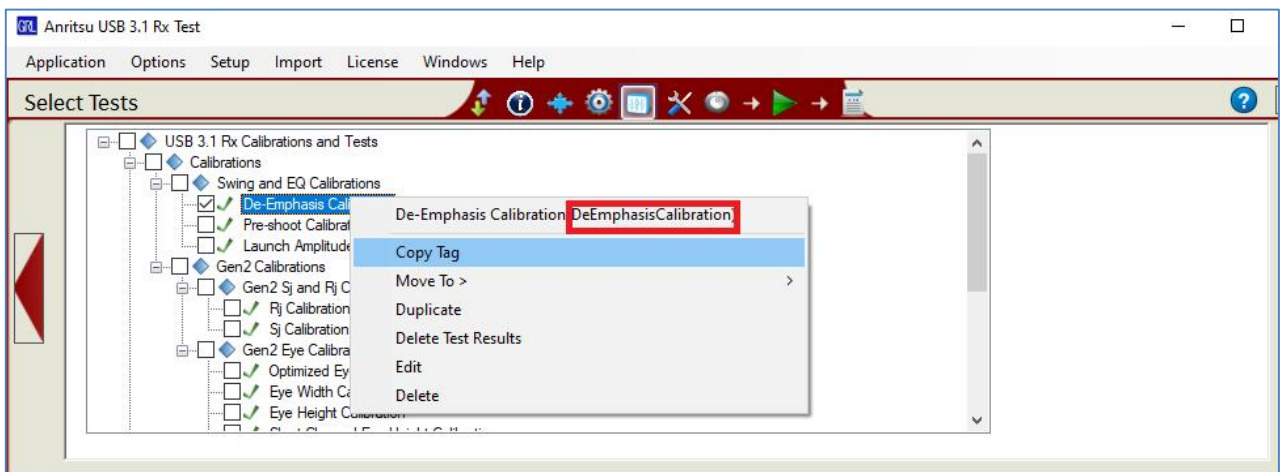


FIGURE 66. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– FRACTION CODE COMMAND #3

d) `FWAppObj.SendCommand("RunTests", None);`

- Executes the "Run Tests" operation as follows:

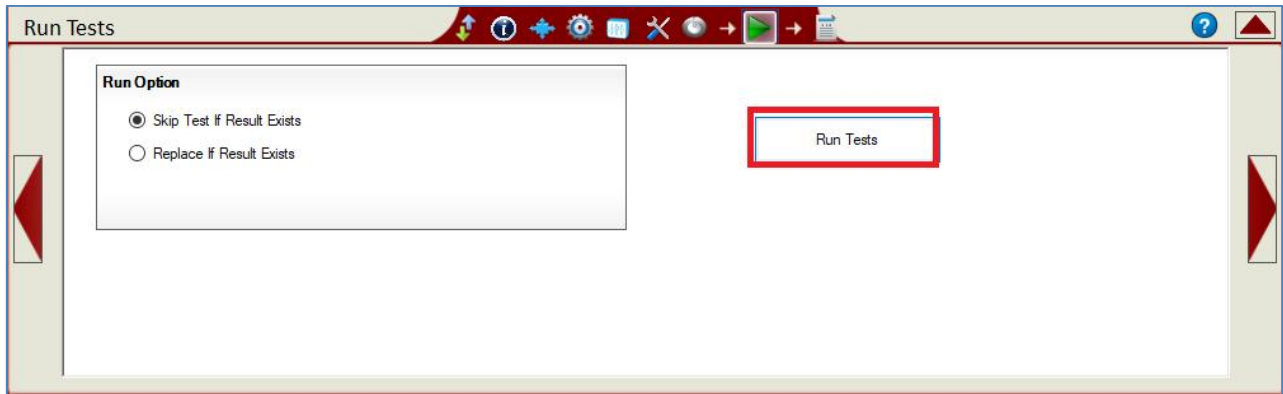


FIGURE 67. USING GRL VISUAL SCRIPTING TOOL EXAMPLE– FRACTION CODE COMMAND #4

7 Using Predefined Modules

The GRL KayaQ Framework provides some predefined modules for certain tests that have common a test flow and are frequently used.

Two predefined modules are available with the GRL KayaQ Framework license as follows:

- **Standard Measurement Step Module**– A module that sums up the test flow of a typical waveform measurement using a oscilloscope.
- **Calibration Step Module**– A module that sums up a test flow of a typical calibration procedure of a BERT parameter.

Sample module projects are provided in the C:\GRL\GRL KayaQ Framework Test\Sample Projects\ folder.

The functionalities of each test defined in the sample projects are:

- Rise Time– Loads a Scope setup file that prepares the Scope for rise time measurement and then reads the rise time measurement.
- RJ Calibration– Calibrates the RJ parameter of a BERT to a specified target.
- SJ Calibration– Calibrates the SJ parameter of a BERT to a specified target.
- Eye Height Calibration– Adjusts the BERT's level and calibrates the level until the Eye height of the signal reaches the specified target.
- ISI Calibration– Calibrates the ISI of the ISI generator to specified limits.

7.1 Using Standard Measurement Step Module

When validating a device, there are usually a list of measurements to be performed. The Standard Measurement Step module replicates the following manual steps example to perform Scope measurements in a flowchart-like GUI:

- a) The user configures the BERT to output signals to be measured and saves the configuration file as "Rise Time config". *(This step is optional, which is only required if the BERT signal is to be measured).*
- b) The user then configures the Scope to display Function1 (Channel1 – Channel3) to perform a measurement e.g., rise time measurement and saves the setup as "ScopeRiseTimeSetup".
- c) The user then fine tunes the waveform by manually adjusting the scales of Channel1, Channel3 and Function1.
- d) The user then acquires the signal multiple times and reads the measurements from the Scope.



The manual steps as defined above can be applied to any measurements such as SJ, RJ, Eye measurements, Vpp, etc. as long as the user can save the Scope setup configurations to perform these measurements.

All of these step measurements can be easily implemented by using the Standard Measurement Step Module flowchart-like GUI. Below shows the equivalent mapping to the steps by using the GUI:

- The user configures the BERT and after that clicks on the “Save” button. Optionally the user can override the test pattern used by providing the appropriate pattern file.
- The user configures the Scope setups and then click on the “Save” button.
- The user keys in the Scope channel used and needing to be auto scaled. In this case, “CH1,CH3,MATH1” to indicate Channel1, Channel2 and Math Function1 respectively.
- No action is then required as waveforms will be acquired by default using the module.
- The user writes a little bit of code to pass back the measurements to the framework.

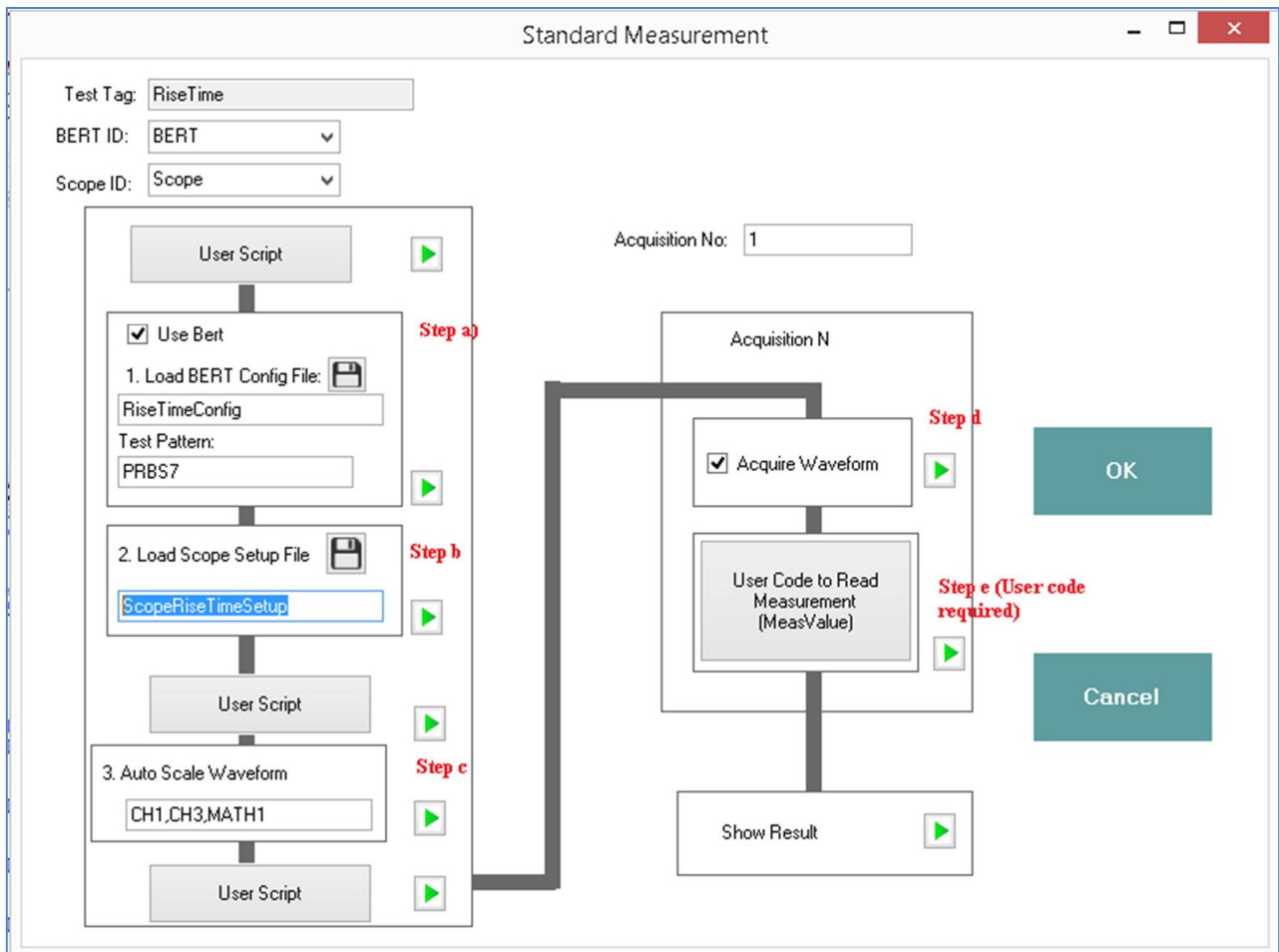


FIGURE 68. USING PREDEFINED MODULES EXAMPLE– STANDARD MEASUREMENT STEP MODULE GUI

Referring to Step e) above where the user can write the user code to read the measurements, the user just needs to write the code to retrieve the Scope measurements and then pass back to the framework by a hard-coded variable “MeasValue” through the module.

In the code, the user will need to first read the measurements from the Scope. For instance, the Python code can be:

```
# Write a script sequence that reads the measurement you need.
Scope = Equipment.GetEquipment("Scope")
Result = Scope.Query("DPOJET:MEAS1:RESULTS:CURRENTACQ:MEAN?")

#MeasValue is the hard-coded variable used to pass result back to
Framework
FWAppObj.SetVariable("MeasValue", Result );
```

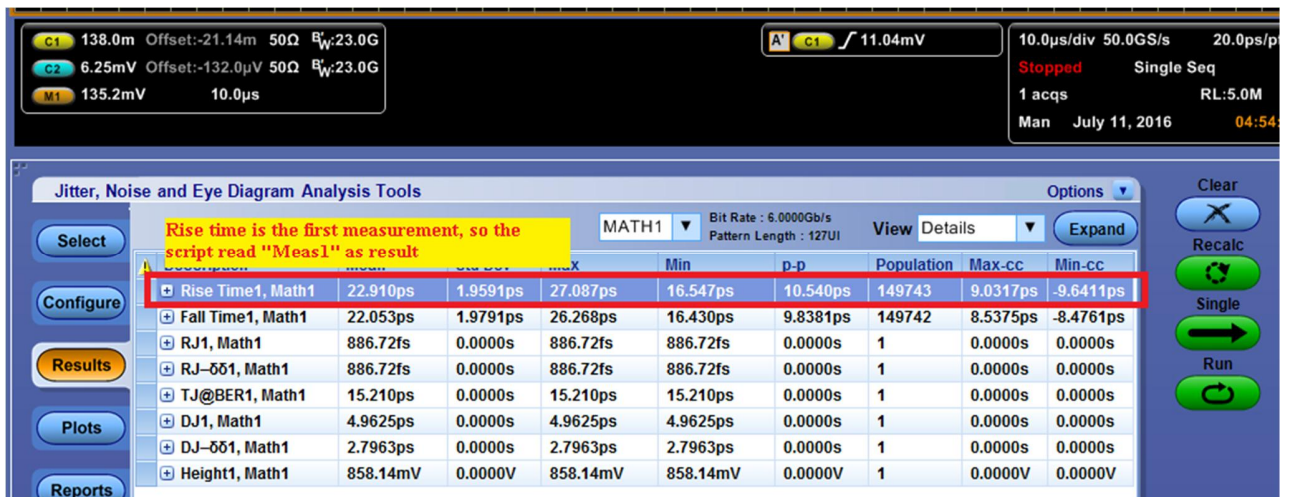


FIGURE 69. USING PREDEFINED MODULES EXAMPLE– WRITE CODE TO READ SCOPE MEASUREMENTS

8 Appendix A: Sample Projects

A “DemoApplication” sample project is available and installed here:
C:\GRL\GRLFramework\Sample Projects\DemoApplication.

The following scripts are provided:

- **ControlChanged.py**– A script tied to the **OnControlChangedEvent** event. The script shows an example of how to change the states of user UI control elements such as Text box, Combo box and Check box in the Framework.
- **OnRunStarted.py**– A script tied to the **OnRunStart** event that simply sets an Internal Global “Global Testing” variable during the test. The internal variable values remain unchanged throughout the test session.
- **DemoTest.py**– A script that shows how to send a command to the equipment setup, and how to write a test sequence.
- **TestMessageBox.py**– A script that shows how to display a message box dialog.
- **ErrorHandling.py**– A script that shows how to handle errors and check if the “user pressing the stop button” event is triggered.

END_OF_DOCUMENT