TekExpress®
Automated USB 3.0 Receiver Compliance and Margin Test (USB-RMT) Solutions

Online Help

Tektronix

TekExpress®
Automated USB 3.0 Receiver Compliance and Margin Test (USB-RMT) Solutions

Online Help

Tektronix

Contacting Tektronix

Tektronix, Inc.
14150 SW Karl Braun Drive
P.O. Box 500
Beaverton, OR 97077
USA

For product information, sales, service, and technical support:
- In North America, call 1-800-833-9200.
- Worldwide, visit www.tektronix.com to find contacts in your area.

Table of Contents

TekExpress Programmatic Interface

Troubleshooting

Reference

Index

General Safety Summary

Review the following safety precautions to avoid injury and prevent damage to this product or any products connected to it.

To avoid potential hazards, use this product only as specified.

Only qualified personnel should perform service procedures.

While using this product, you may need to access other parts of a larger system. Read the safety sections of the other component manuals for warnings and cautions related to operating the system.

To Avoid Fire or Personal Injury

Connect and disconnect properly. Connect the probe output to the measurement instrument before connecting the probe to the circuit under test. Connect the probe reference lead to the circuit under test before connecting the probe input. Disconnect the probe input and the probe reference lead from the circuit under test before disconnecting the probe from the measurement instrument.

Observe all terminal ratings. To avoid fire or shock hazard, observe all ratings and markings on the product. Consult the product manual for further ratings information before making connections to the product.

Do not operate without covers. Do not operate this product with covers or panels removed.

Do not operate with suspected failures. If you suspect that there is damage to this product, have it inspected by qualified service personnel.

Avoid exposed circuitry. Do not touch exposed connections and components when power is present.

Terms in This Manual

These terms may appear in this manual:

⚠ ***WARNING.** Warning statements identify conditions or practices that could result in injury or loss of life.*

⚠ ***CAUTION.** Caution statements identify conditions or practices that could result in damage to this product or other property.*

TekExpress Automated Solutions

Using Online Help

Select Help from the menu to open the help file. You can also find an electronic copy of the help file in the Documents directory on the 063-4068-XX DVD.

Contents tab – Organizes the Help into book-like sections. Select a book icon to open a section; select any of the topics listed under the book.

Index tab – Enables you to scroll a list of alphabetical keywords. Select the topic of interest to bring up the appropriate help page.

Search tab – Allows a text-based search of the online help content.

Follow these steps:

1. Type the word or phrase you want to find in the search box. If the word or phrase is not found, try the Index tab.

2. Choose a topic in the lower box, and then select the Display button.

General Help Functions:

- Select the Print button from the Help topics menu bar to print a topic.

- To return to the previous window, select the Back button.

- Use hyperlinks to jump from one topic to another.

- If the back button is grayed out or a jump is not available, choose the Help Topics button to return to the originating help folder.

Related Documentation

Technical Specification Documents

http://www.tek.com/Measurement/applications/serial_data/sata.html

Conventions

Online Help uses the following conventions:

- The term "USB" refers to Universal Serial Bus.

- The term "DUT" is an abbreviation for Device Under Test.

- The term "select" is a generic term that applies to the two mechanical methods of choosing an option: using a mouse or using the touch screen.

Table 1: Icon descriptions

Icon	Meaning
	This icon identifies important information.
	This icon identifies conditions or practices that could result in loss of data.
	This icon identifies additional information that will help you use the application more efficiently.

Technical Support

Tektronix values your feedback on our products. To help us serve you better, please send us your suggestions, ideas, or comments on your application, signal generator, or oscilloscope.

When you contact Tektronix Technical Support, please include the following information (be as specific as possible):

General Information

- All instrument model numbers.
- Hardware options, if any.
- Probes used.
- Your name, company, mailing address, phone number, FAX number.
- Please indicate if you would like to be contacted by Tektronix about your suggestion or comments.

Application Specific Information

- Software version number.
- Description of the problem such that technical support can duplicate the problem.
- If possible, save the setup files for all the instruments used and the application.
- If possible, save the TekExpress setup files, log.xml and status messages text file.
- If possible, save the waveform on which you are performing the measurement as a .wfm file.

Forward the information to technical support using one of these methods:

- E-mail – techsupport@tektronix.com
- FAX – (503) 627-5695

What Is New in This Release

The newest release of TekExpress USB-RMT (v1.2.0) includes the following features:

- Support for the AWG7122C and AWG7102 Arbitrary Waveform Generators

- Support for 8 GHz oscilloscopes

- Support for the C models of MSO/DPO/DSA70000 Series oscilloscopes

- Support for Windows 7

- Improved handling of 'Waveform Creation option"

    - Use Pre-created waveforms and create waveforms if not available

    - Use Pre-created waveforms (Normative test uses provided *.awg files. Preload AWG files or use pre-created waveforms (compliance mode)

    - Use Pre-created waveforms (Informative test checks for file. If file is not present, a message is displayed and the test continues to execute.

- CP1 clean pattern for Manual Loopback Verification

- User defined AWG sequence files for Loopback Initialization

- Calibration value for compliance test

- User Friendly data entry for Informative test with a default preset scan

Previous releases of TekExpress USB-RMT included the following features:

- Provides customers with a complete solution for USB 3.0 receiver verification, characterization, debug, and compliance.

- Fully automates USB 3.0 Receiver Compliance and Margin testing.

- Automatically creates all required waveforms to perform receiver testing.

- Automatically negotiates the device into loopback mode.

- Provides real-time update of the jitter tolerance curve after execution of each test point.

- Generates a comprehensive report with test results and signal characteristics

- Provides complete integration with the Ellisys Error Detector and Oscilloscope Error Detector

Accessories

About the Test Fixture

For host testing.   (1) TF-USB3-A-R, (1) TF-USB3-B-R.

For device testing.   (2) TF-USB3-A-R (includes short USB 3.0 Cable), (1) TF-USB3-B-R.

For error detection.   External Error Detector Ellisys USB Explorer 280 (Analyzer and Traffic Generator for SuperSpeed USB3.0), or Oscilloscope Error Detector.

Minimum System Requirements

The minimum system requirements for a PC or Microsoft Windows-based oscilloscope to run TekExpress USB-RMT are as follows:

NOTE. TekExpress USB-RMT does not run on a Tektronix AWG7102, AWG7122B, or AWG7122C.

Table 2: System requirements

Processor	Pentium 4/M or equivalent processor.
Operating System	Microsoft Windows XP Service Pack 2.
Memory	512 MB of memory.
Hard Disk	Approximately 2 GB of available hard-disk space for the recommended installation, which includes full TekExpress installation and distributed components.
Drive	DVD drive.
Display	Super VGA resolution or higher video adapter (800x600 minimum video resolution for small fonts or 1024x768 minimum video resolution for large fonts). The application is best viewed at 96 dpi display settings [1].
Software	■ Microsoft Internet Explorer 7.0 or later. ■ Microsoft Excel 2002 or above. ■ Adobe Reader 6.0 or equivalent software for viewing portable document format (PDF) files.
Other Devices	■ AWG7122B or AWG7122C with firmware for both hardware and software channel emulation, or AWG7102 with firmware for hardware channel emulation only. ■ Microsoft compatible mouse or compatible pointing device. ■ Two USB ports (four USB ports preferred). ■ PCI-GPIB or equivalent interface for instrument connectivity [2].

[1] If TekExpress is running on an instrument having a video resolution lower than 800x600 (for example, a sampling oscilloscope), it is recommended to connect and activate a secondary monitor before launching the application.

[2] If external devices such as USB-GPIB or equivalent are used for instrument connectivity, ensure to enable the Talker Listener utility in the GPIB menu of the Tektronix DPO/DSA Series oscilloscope.

Application Directories and Usage

TekExpress USB RMT is installed in the following directory path, depending on the Windows operating system.

- For Windows 7:

  - C:\Program Files (x86)\Tektronix\TekExpress\TekExpress USB RMT

- For Windows XP and Windows XP-Embedded:

  - C:\Program Files \Tektronix\TekExpress\TekExpress USB RMT

The application directory and associated files are organized as follows:
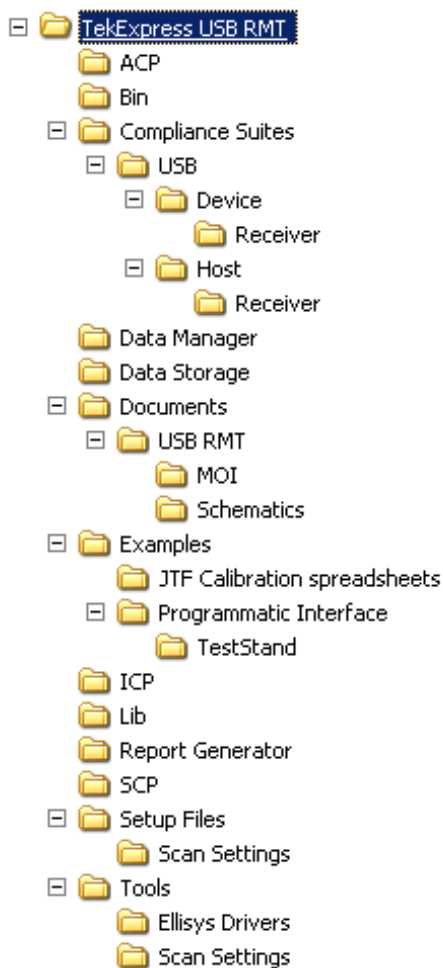
The following table lists the default directory names and their usage:

Table 3: Default directory names and their usage

Directory names	Usage
InstallDir\TekExpress	Contains the TekExpress application and associated files.
\TekExpress\TekExpress USB RMT	Contains files specific to TekExpress USB- RMT.
\TekExpress USB RMT\Compliance Suites	Contains compliance specific sequence files. The folders under this directory represent the devices to be tested.
\TekExpress USB RMT\Compliance Suites\USB	Includes the Device folder.
\TekExpress USB RMT\SCP \TekExpress USB RMT\ICP	Includes the TekExpress instrument and application specific interface libraries.
\TekExpress USB RMT\Data Manager \TekExpress USB RMT\Data Storage \TekExpress USB RMT\Report Generator	Includes the TekExpress result management specific libraries.
\TekExpress USB RMT\Tools\Ellisys Drivers	Includes the driver files for the Ellisys Protocol Analyzer Hardware.
\TekExpress USB RMT\Tools\Scan Settings	Includes the default custom jitter profile template files.
\TekExpress USB RMT\Documents	Includes the Method of Implementation documents and technical documentation for the application.
\TekExpress USB RMT\Bin \TekExpress USB RMT\Lib \TekExpress USB RMT\Tools	Includes the TekExpress Miscellaneous libraries.

File Name Extensions

The software uses the following file name extensions:

File name extension	Description
.awg	The AWG7XXX sequence files.
.pdf	The PDF file that how to implement the test method.
.s4p	The S-Parameter files used to create waveforms when Channel Emulation is set to Software
.seq	The test sequence file.
.TekX	The session file is saved in this format.
.wfm	The waveform files created and kept in the AWG7XXX.
.xml	The encrypted XML file that contains the test specific configuration information. The log file extension is also xml.

How to Activate the License

Follow the steps below to activate the license:

1. Click **Help > About TekExpress** to view the license information.

2. Click the **View Version Details** link to check the version numbers of the installed test suites.
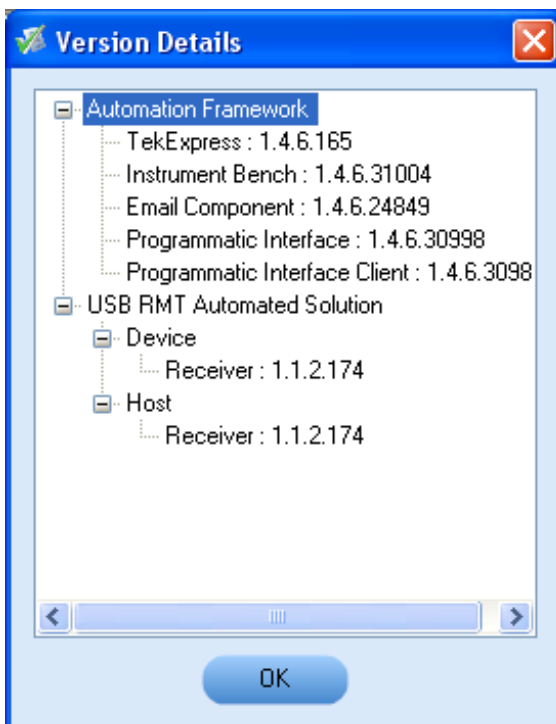
3. Click the **View End-User License Agreement** link to open the following Tektronix Software License Agreement window. Click **Print** to print the License Agreement.
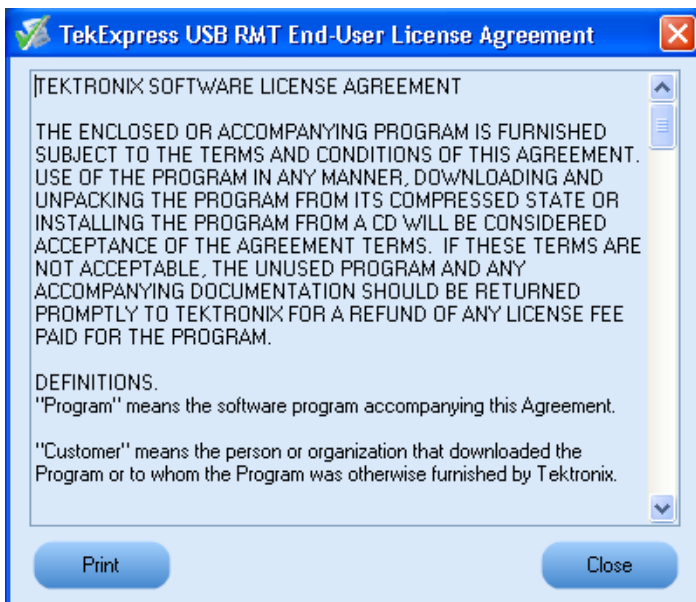
4. Click **License Info** to view the available software options. This window shows the license key and the various options with their status (active or inactive) with the current license key.

5. If you are activating the license for the first time, the license key field is empty. To activate the license, connect the USB dongle to your computer, enter the license key provided in the license key certificate, and click **Activate**. If the activation is successful, a sign is displayed next to the license key field.
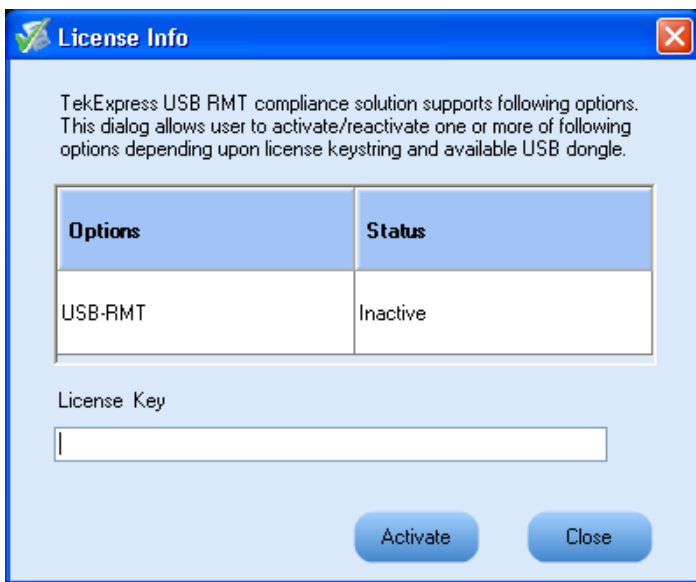
6. If you are reactivating the license, click **Reactivate**, enter the new license key and click **Activate**.

Before You Click Run

After you first launch TekExpress, it creates the following folders on your computer:

- `\My Documents\`**`My TekExpress`**

NOTE. *Ensure that the "My TekExpress" folder has read and write access.*

NOTE. *If a user with a different Windows login ID launches TekExpress, a new `My TekExpress` folder is created under that user's `My Documents` folder*

- `\My Documents\My TekExpress\`**`USB RMT`**.
- `\My Documents\My TekExpress\USB RMT\`**`Untitled Session`**. Every time the `TekExpress USBRMT.exe` is launched a `Untitled Session` folder is created under `USB RMT` folder. The `Untitled Session` folder is deleted when you exit TekExpress.

⚠ **CAUTION.** *Each session has multiple files associated with it. Do not modify any of the session files and/or folders as this may result in loss of data or corrupted session files.*

- The `My TekExpress` folder is created as a shared folder with share name as `<domain><user ID> My TekExpress` (or if the PC is not connected to domain then share name is `<Computer name><user ID> My TekExpress`).
- The above shared folder is mapped as X: (X drive) on to the PC where TekExpress is running.

NOTE. *If X drive is mapped to any other shared folder, TekExpress will display a Warning message window asking to disconnect the X: drive manually.*

Do the following before you click Run:

NOTE. *Make sure to enable the network connectivity on the PC running the TekExpress application.*

---

1. Map (see page 13) the shared `My TekExpress` folder as X: (X drive) on all the instruments used in test setup running Microsoft Windows Operating System. This share folder is used to save the waveform files or any other file transfer operations.

2. Right-click on the `My TekExpress` folder and open the **Properties** dialog box. Select the**General** tab and then **Advanced**. In the **Advanced Attributes** window, ensure that the option **Encrypt contents to secure data** is NOT selected. Click here (see page 14) to view the picture.

3. Ensure that the USB RMT setup files provided with TekExpress DVD are available on the respective instruments. For more details, refer to the `ReadmeFirst.txt` located in the `USB RMT Setup Files` folder on the TekExpress DVD.

4. Ensure that all the required instruments are properly warmed up, and that you do a Signal Path Compensation (SPC) (see page 14). Add calibration of the AWG.

Mapping My TekExpress Folder

To map the My TekExpress folder on the instruments, follow the steps below:

1. Open Windows Explorer.

2. From the Windows Explorer menu, select **Tools > Map Network drive**.

3. Select the Drive letter as X: (if there is any previous connection on X:, disconnect it first through **Tools > Disconnect Network drive** menu of Windows Explorer).

4. In the Folder field, enter remote `My TekExpress` folder path (for example, \\192.158.97.65\ John's My TekExpress)

5. Determine the IP address of the PC where "My TekExpress" folder exists by doing the following:

   - Select **Start > Run** menu on the PC where My TekExpress folder exists.

   - Enter `cmd` and click **Enter**.

   - At command prompt, type `ipconfig`.
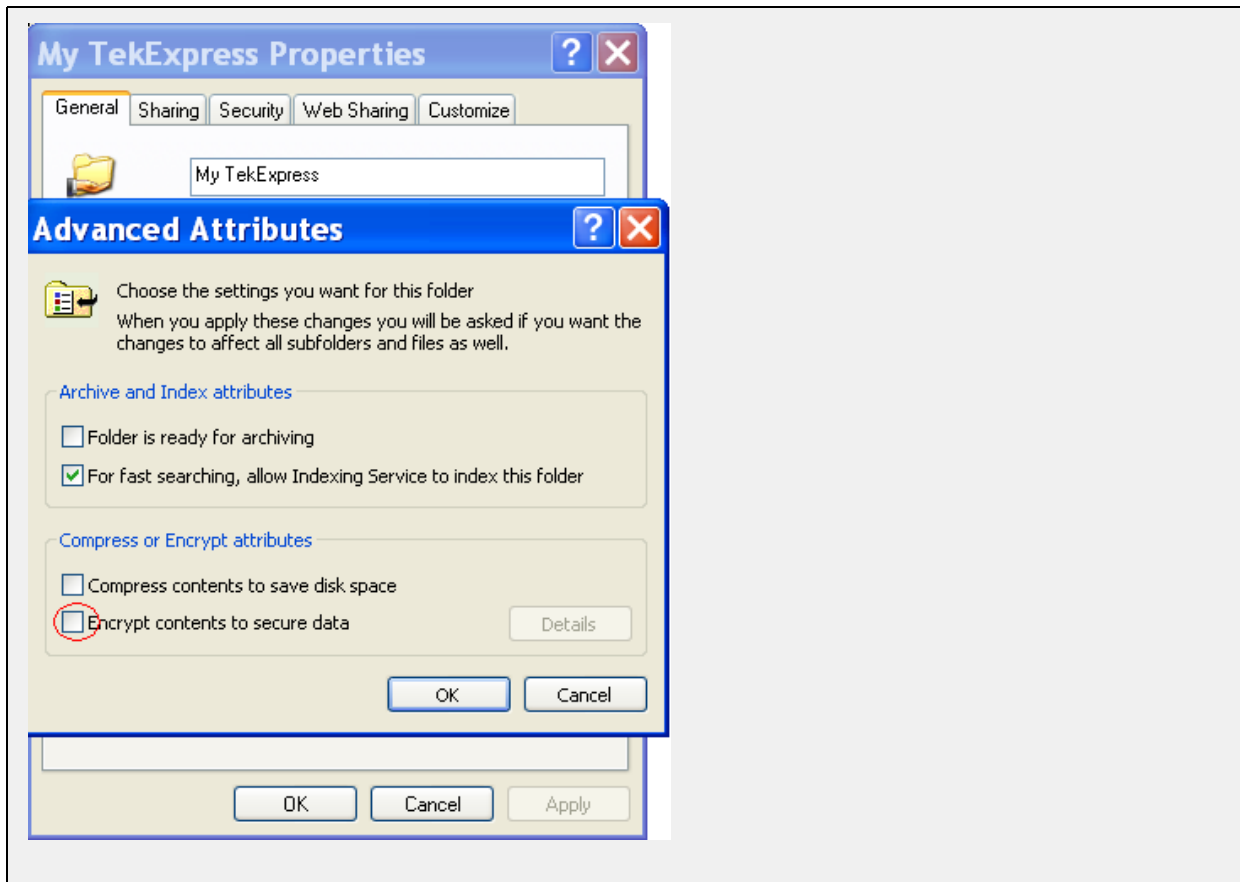
Find SPC by following the steps below (only needed if using an oscilloscope error detector):

1. On the oscilloscope main menu, click **Utilities** menu.

2. Click **Instrument Calibration** option.

TekExpress Application Overview

TekExpress is the Tektronix Compliance Test Automation Framework, developed to support current and future test automation needs of customers. Developed using National Instruments' TestStand, TekExpress leverages on the capabilities of Microsoft .NET framework. It is a highly modular architecture that enables deploying automated test solutions for various serial standards in a relatively short time. The USB receiver solution supports test automation in the TekExpress framework providing customers with a complete solution for USB 3.0 receiver verification, characterization, debug, and compliance.

Starting the Application

The application uses a USB dongle that contains the license key. This dongle must be present on the PC or the instrument hosting the TekExpress application.

The application also checks for a file, called `Resources.xml`, located in My TekExpress folder. If this file is not found, instrument discovery is performed before launching TekExpress. The `Resources.xml` file contains information regarding instruments available on network.

When the application starts, it checks for the appropriate license key. If the valid license key is not present, the application switches to the "Demo" mode. If the application fails to detect the dongle, it continues to run in Demo mode.

To start the application, do one of the following:

■ Click **Start > Programs > Tektronix > TekExpress > TekExpress USB RMT**. Other applications follow similar pattern.

■ Double-click the icon ![icon] on the desktop.

■ If you have previously saved a session, double-click the session file stored under `My TekExpress\USB-RMT`.

When the application is launched it displays the splash screen providing launch information. The application also checks for the presence and validity of the USB dongle.

NOTE. *If the application was not terminated properly during the last use, a dialog box asks to recall the previously unsaved session.*

Resizing the Application Window

■ To minimize the application, click on the application title bar. To restore the application to its previous size, select in the Windows task bar.

■ To maximize the application, click . To restore it to previous size, click on the application title bar.

Exiting the Application

To exit the application, do one of the following:

■ Click **File > Exit**.

■ Click on the application title bar.

Global Controls

The menus and controls that appear outside the individual tabs are called "Global Controls". These are used to specify the devices to be tested.
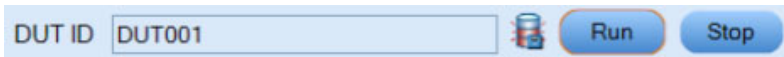
Table 4: Controls and functions

Control name	Function
DUT	The device ID is specified at the global level and the information is stored in the default location for all data files. This field cannot be empty and does not allow these special characters (.,..,...,\,/:?"<>|*). The maximum length of characters allowed is 32.
	This indicates when the disk memory goes below ~1 GB of available space on the device running TekExpress.
	You are able to run, pause, resume and stop the tests.

File Menu

Click **File** on the application menu bar.
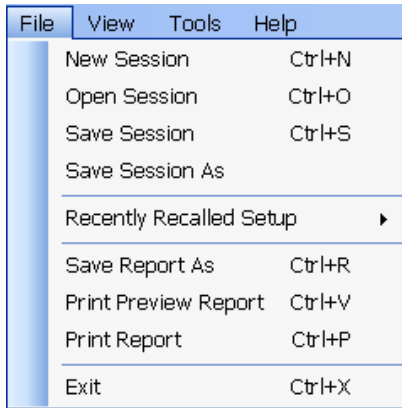
The File menu has the following selections:

Menu	Function
New Session	Starts a default session of TekExpress.
Open Session	Opens a saved session.
Save Session	Saves the session.
Save Session As	Saves a session in a different name.
Recently Recalled Setup	Lists all the recent and previously recalled setup files.
Save Report As	Saves the report in user specified location.
Print Preview Report	Previews the report before printing.
Print Report	Opens the Windows "Print" dialog box.
Exit	Closes the application.

View Menu

Click **View** on the application menu bar.
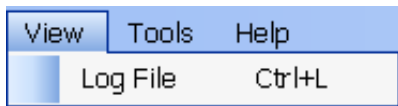
The View menu has the following selections:

Menu	Function
Log File	Opens the log (log.xml) file in the default viewer.

Tools Menu

Click **Tools** on the application menu bar.
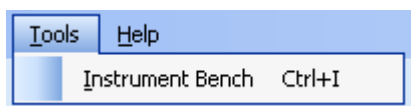
The Tools menu has the following selections:

Menu	Function
Instrument Bench (see page 19)	Opens a dialog box showing the list of instruments attached to the test setup.

Tools > Instrument Bench

The Instrument Bench window shows the list of VISA and Non-VISA resources found on different interfaces/connections. It serves two purposes at the launch of TekExpress:

NOTE. *The Ellisys analyzer is a Non-VISA Resource and should be checked in order for TekExpress to recognize the Ellisys analyzer.*

- ■ Discovers the connected instruments.
- ■ Confirms the instrument connection setup.

When you click **Tools > Instrument Bench**, the following dialog box is displayed:
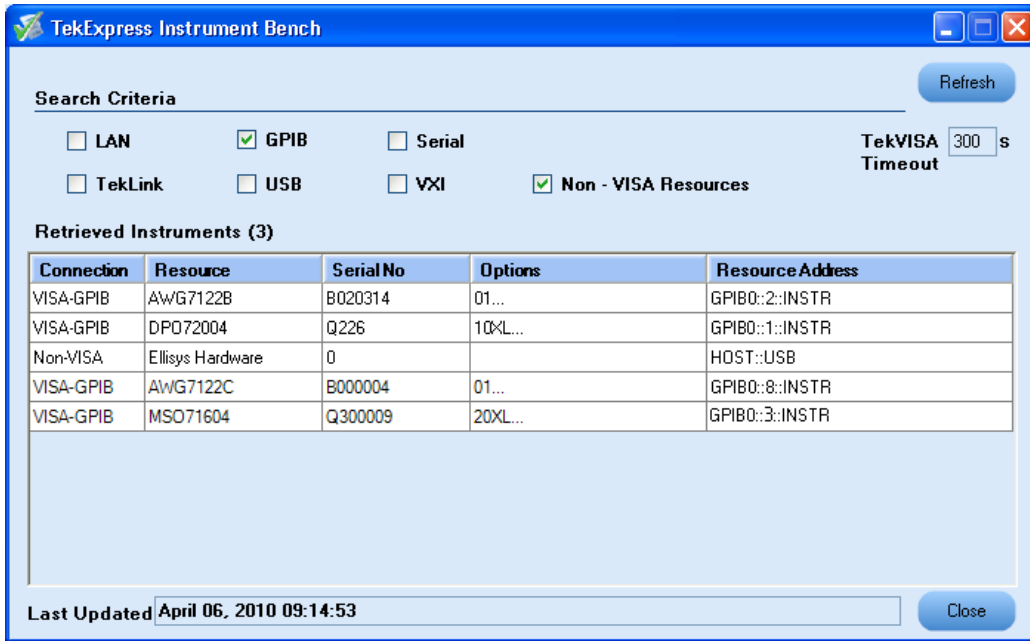
- ■ **Search Criteria**: The various connections on which you can search. **Non-VISA Resources** are the instruments that cannot be searched using TekVISA.

- ■ **Retrieved Instruments**: Displays the count and details of instruments that were discovered.

- ■ **Last Updated**: Displays the time when the last time search was performed.

- ■ **TekVISA Refresh Timeout (Seconds)**: This time out specifies the maximum time that TekExpress can wait for TekVISA update.

NOTE. TekExpress uses TekVISA for instrument search. Ensure that TekVISA is running on your system before you refresh the instrument bench window.
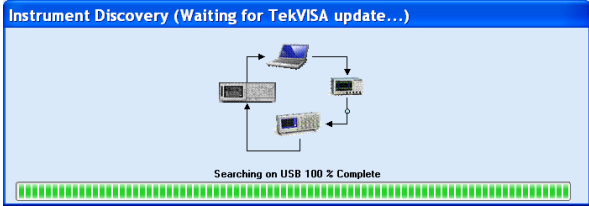
Table 5: Retrieved Resources properties in the Instrument Bench window

Title	Description
Connection	Shows the type of connection with the instrument.
Resource	Shows the name of the resource.
Serial Number	Shows the serial number of the resource.
Options	Shows the options available on the instrument. [1]
Resource Address	Shows IP Address/Port number of the resource.

[1]   The option column displays the options that fit in the field. To view complete options on the instrument, move the mouse cursor over the option.

Table 6: Button controls on Instrument Bench dialog box

Button	Function
Refresh	The application searches on the selected connection for resources. While searching resources it shows the Instrument Bench discovery window. The Discovery window shows the connection currently being scanned and the percentage of task completed.
Close	Closes the dialog box.

Help Menu

Click **Help** on the application menu bar.
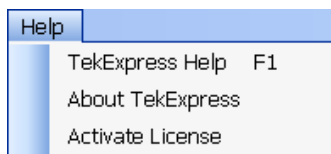
The Help menu has the following selections:

Selection	Description or Function
TekExpress Help	Displays TekExpress Help (F1).
About TekExpress (see page 10)	Displays application details such as software name, version number and copyright.
Activate License	Displays available software options and also about license activation.

Select the Test(s)

The application supports the following DUT types for Receiver Compliance and Characterization testing compliance.

- Device

- Host

Use the Select panel to select either a compliance test or a margin test (which allows the sweeping of SJ frequencies) to configure and run.
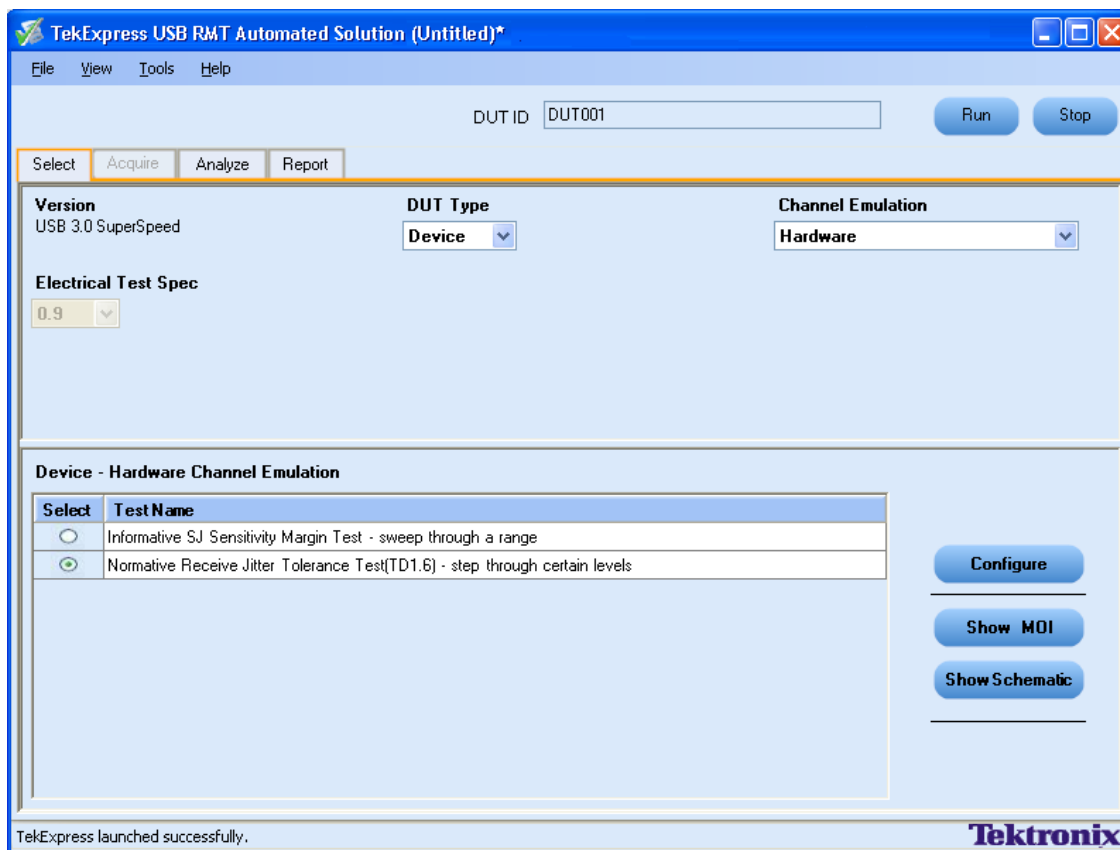
The Select panel provides the following functions:

Version

Specifies the USB version number.

DUT Type

Specify either Device or Host as DUT. This specification will ensure that the appropriate channel model is used when using software channel emulation.

Channel Emulation

A channel emulator duplicates (provides an emulation of) the function of the actual physical channel and cable, so that the emulated channel behaves like the actual physical channel. The default is to use the USB 3 compliance channels. Select one of the following:

Hardware	Select this option to test the application using a USB cable instead of S-parameter files.
Software	Select this option to emulate the channel using the S-parameter files.

Electrical Test Specification

Specifies the version number of the Electrical Compliance Test Specification (CTS) document. The CTS document provides the compliance criteria and test descriptions for SuperSpeed USB devices, hubs and host controllers that conform to the Universal Serial Bus 3.0 specification. These criteria address the electrical requirements for a SuperSpeed physical layer design.

All signal impairments (such as De-emphasis, SSC Profile, RJ, and others) are automatically configured based on the version of the CTS selected. Calibration of the signal impairments is required to ensure that the output of the AWG meets the specifications.

Device Channel Emulation

This depicts the type of system configuration.

- **Select**: Includes/Excludes any test for analysis.

- **Test Name**: Displays the name of the test.

Once you select a row, the following options are available:

Table 7: Button controls on the Select panel

Button	Description
Configure	Opens the configuration panel for the selected test.
Show MOI	Opens the PDF of method of implementation (MOI) for the selected test.
Show Schematic	Opens the schematic for the selected test. This is useful if you want to verify the test setup before running the test.

Configure and Run the Test(s)

The configuration panel is used to create, view, and edit the parameters associated with the acquisition and the analysis of the selected test.
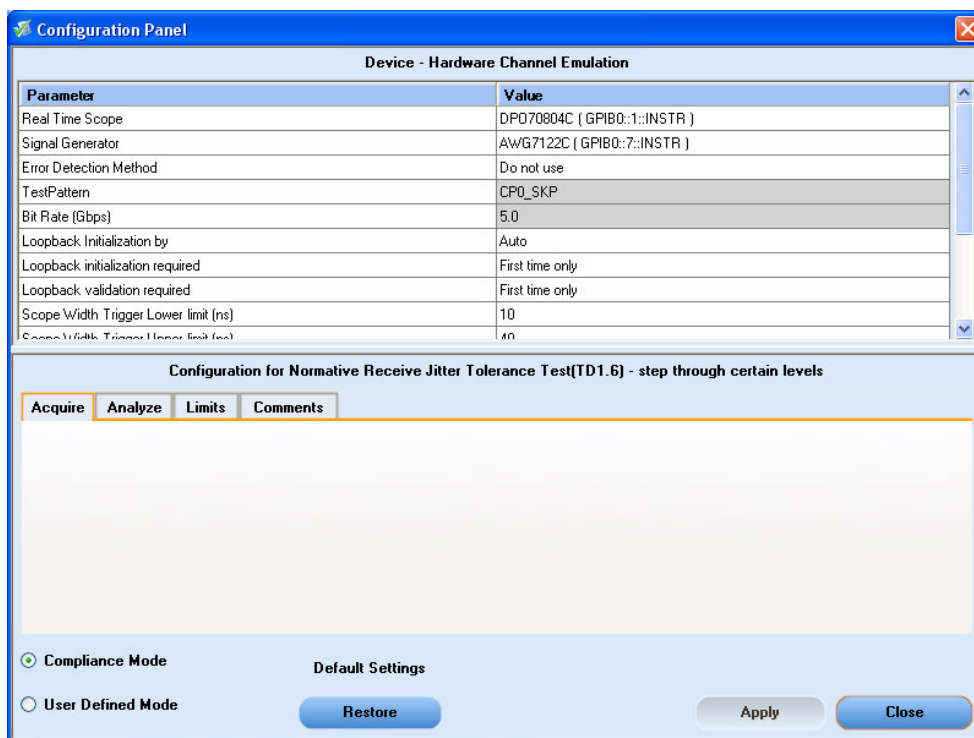
You have the following options:

■ Choose between running the tests in a Compliance or User Defined mode. If you wish to change to User Defined mode, you are prompted with a message "If you make changes to a test, the test may no longer be compliant". In User-defined mode you can change parameters such as the SJ Frequency and amplitude, SSC profile, the amount of RJ and other parameters.

■ Store compliance mode values.

- Change the parameters associated with analysis configuration.

- Specify the error detection method.

- Specify Loopback initialization (see page 27) and validation.

- Change the test limits (allowed number of bit/symbol errors).

The upper half of the Configure panel has general parameters that are common for all the tests under the selected test suite that are editable. The lower half of the Configure panel has test specific parameters.

NOTE. If any of the test parameters are grayed, it means that these parameters cannot be modified in compliance mode. When you switch to user-defined mode, these parameters are editable.

Table 8: Test parameters

Parameters to configure	Description
Acquire	Typically, it shows any acquisition parameters, but for RMT testing no acquire parameter is applicable.
Analyze	Shows the various parameters related to analysis of a selected test. You can include/exclude various jitter parameters based on amplitude and frequency. Double-click on the row to include/exclude the parameter from the drop-down menu. You can also set the following parameters: ■ Specify whether to create waveforms or use prerecorded waveforms for signal generation. ■ Specify whether or not to save the created waveforms.
Limits	Applies to a specific test. You can view the Maximum Allowable number of errors for error detection. The application automatically detects the number of errors for each test point, if error detector hardware is used. If the errors detected by Ellisys (see page 27) or the Scope Error Detector are more than the specified limit (should be Less Than or Equal To), then the test point is marked as FAIL. If other methods of error detection are used, you need to manually enter the error count.
Comments	You can specify a comment up to 256 characters long for the selected test.
Default settings	
Restore	The default compliance settings will be restored.
Apply	Accepts all changes that you made.
Close	Dismisses the dialog box and does not apply changes.

Click **Run** in the Select panel to run the selected tests.

Ellisys Error Detector

Ellisys USB Explorer 280 Analyzer/Generator is a USB 3.0 symbol error detector. The hardware is identical to the standard Ellisys Super Speed USB 3.0 protocol analyzer/generator. When used with the Tektronix AWG7102, AWG7122B, or AWG7122C high-speed signal generator and automation software from Tektronix, the EX280-CT completes the solution for USB 3.0 receiver testing.

Loopback Initialization

When running the receiver jitter tolerance tests, USB-RMT is configured to automatically put the device into loopback mode per the loopback sequence that is defined in the CTS. Please contact your Tektronix representative if your device requires an alternative method to be put into loopback mode. In the loopback mode, the receiver loops back the data it receives and any difference in the pattern sent from the signal generator and returned from the DUT is counted as an error.

View and Select Connected Instruments

Viewing Connected Instruments

The **Tools > Instrument Bench** menu item is used to discover connected instruments required for the tests. The application uses TekVISA to discover the connected instruments. Once the operation is done, the Instrument Bench dialog box resumes operation and lists the instrument-related details based on the selected search criteria.

NOTE. *When the TekVISA Instrument Manager checks for connected Instruments, the Instrument Bench dialog box does not respond.*

For example, if you select LAN as the search criteria in the Instrument Bench dialog and click Refresh, the TekVISA Instrument Manager checks for the instruments availability over LAN and displays the details of the instrument under **Retrieved Instruments** table.
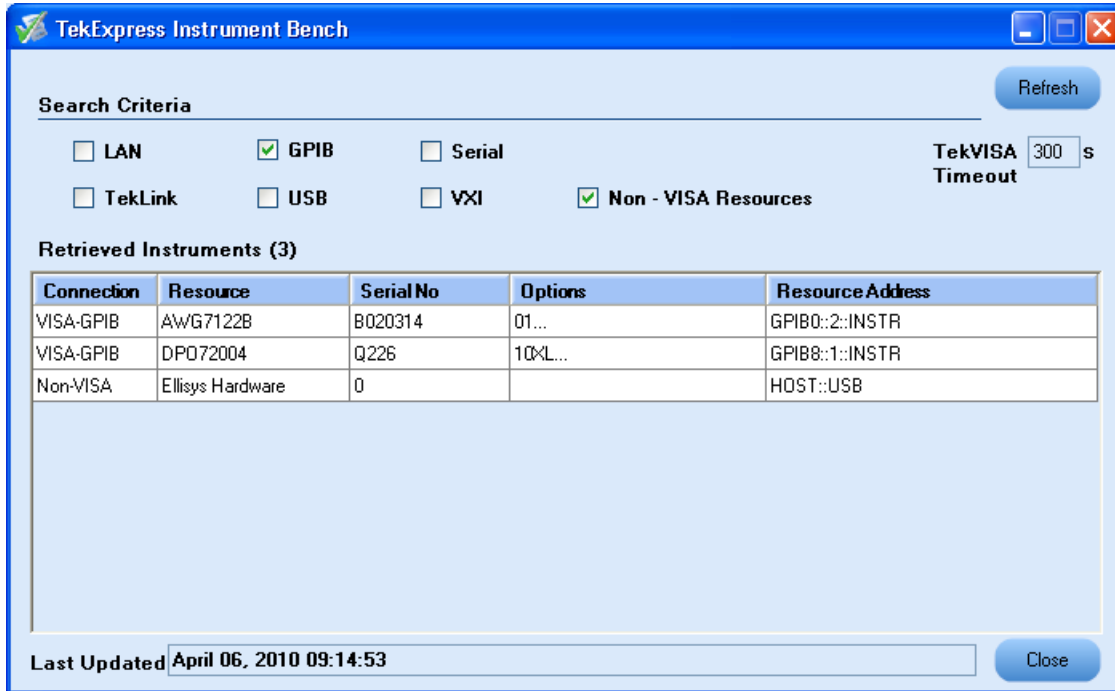
You can provide the time in the **TekVISA Refresh Timeout (Seconds)** field, within which if the TekVISA Instrument Manager does not find the instruments, the TekExpress application resumes the operation.

If you select Non-VISA resources, all the instruments supported by TekExpress but not communicating over the VISA layer can be searched. Example: Ellisys error detector.

Selecting Connected Instruments

You can view the instruments connected in the Configuration panel. The upper half of the panel displays the general parameters for the tests under the selected test suite.

Choose the instruments from the drop-down list as shown in the following figure:
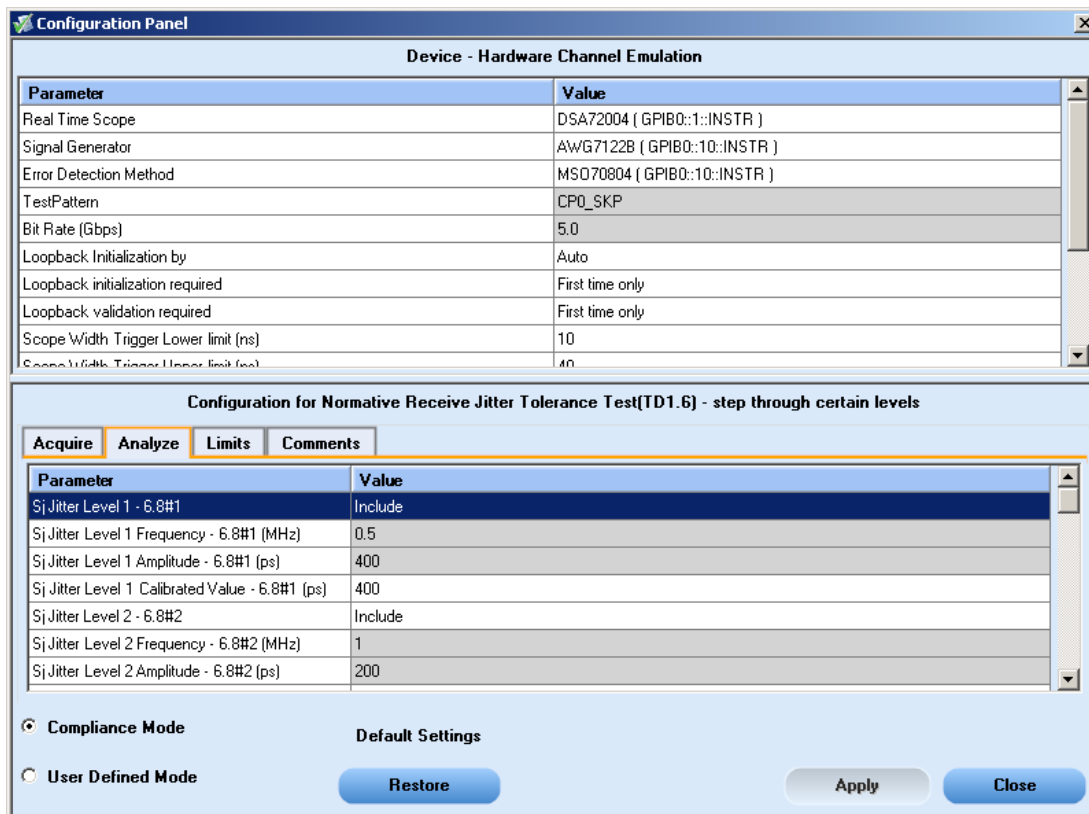
Select the "Do Not Use" option in the drop-down list for those instruments which are optional. For example, if the error detection method is selected as "Do not use", the application prompts you to enter the error count after each test point.

NOTE. *The list of instruments displayed is specific to the selected test suite. It does not show all the connected instruments.*

View the Progress of Analysis

You can view the result summary in the Analyze panel for the selected test. As each test point is executed, the result value is updated.
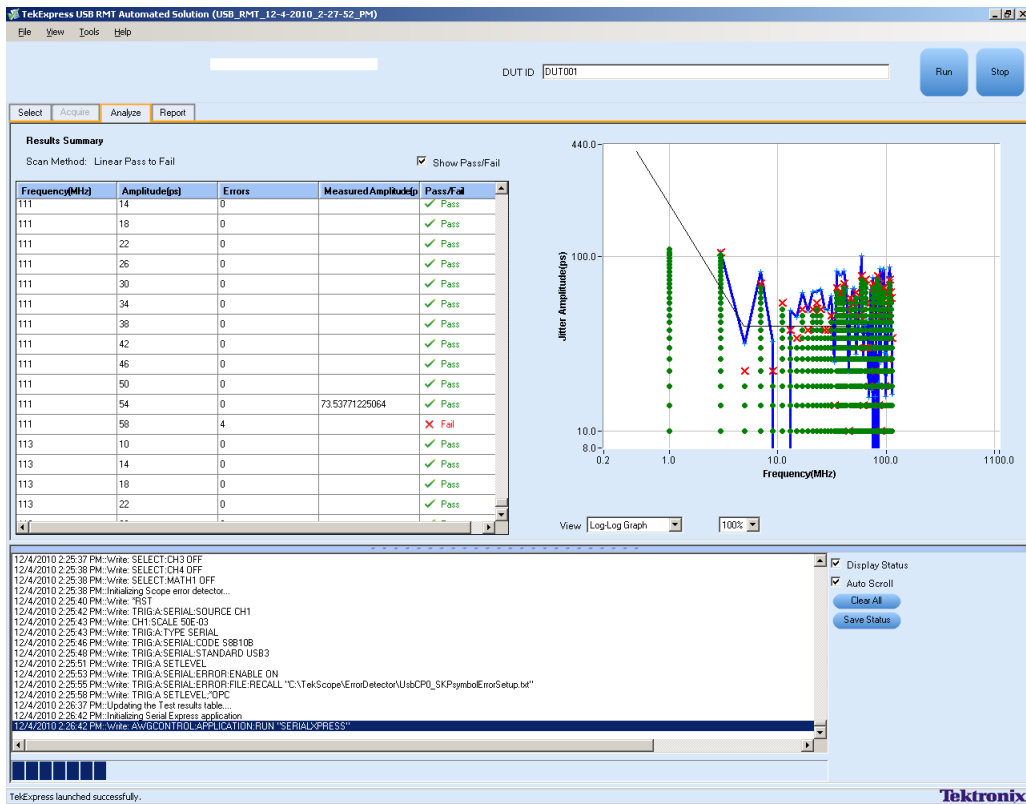
Analysis Table

The application automatically generates all of the required waveforms for each frequency/amplitude as specified using SerialXpress.

Parameter	Description
Frequency	Lists the various SJ frequencies selected in the configure panel after its generation from the AWG.
Amplitude	Lists the various amplitude for each jitter frequency as selected in the configure panel.
Error	Displays any detected errors.
Pass/Fail	Displays Pass/Fail status for each test point.

The test points that have not been executed are shown with a "To be Started" status. A summarized status of the currently running test is shown on the Status Messages panel.

A graph is plotted automatically and updated after the execution of every generated frequency and amplitude. You can view the graph in two modes – and . The graphical view is set to 100%.

The **Status Messages** window timestamps all runtime messages and displays them. Do the following:

- **Display Status**: Enable/Disable status messages.

- **Auto Scroll**: Scroll status messages automatically.

- **Clear All**: Clears all status messages in Status Window.

- **Save Status**: Saves all status messages in text file. Displays a standard save file window and saves the status messages in the user specified file.

NOTE. *The Status Messages window is dockable and can be resized.*

Log-Log Graph

Displays the graph in logarithmic scale for both amplitude and frequency.

Linear Graph

Displays the graph in linear scale for both amplitude and frequency.

View the Report

After the analysis, a report is automatically generated. The report shows the results of the tests, including device information and pass/fail status of each test point.

NOTE. *If a test is aborted, the report is generated for the all completed test points.*

The report displays the DUT, the date and time that was run, and execution time. Test information like the DUT, Execution Time, Compliance Mode (Yes or No), and Overall Test Result (Pass or Fail) are shown. Instrument information like the models, serial numbers, and firmware/software versions are shown. A table displays the Test Name, Measurement Details, Limit values, Test Result, Analysis Execution time and so on.

The Report View Area contains an HTML version of the report template. Select any area of the report and copy it to the clipboard to make it available for other applications.

View Test Related Files

All the test related files for currently selected tests are always saved under `My Documents\My TekExpress\USB-RMT\Untitled Session`.

When you save a session, it is saved with the session name under the path `My Documents\My TekExpress\USB-RMT\SessionName` for future references.

The session that is currently running is stored in the same path as "Untitled" until you save it.

⚠ **WARNING.** *Do not save a session named "Untitled" or "Backup" because these are application-specific files and are deleted when you exit the application.*

A session folder can contain results for more than one DUT, and a DUT folder can contain more than one run data folder marked by date-time stamp as folder name.

Here is an example image of data storage:

USB-RMT Equipment Setup: Device

You need the following equipment to set up the application:

Required equipment	Model supported
Signal Source	Tektronix AWG7102, AWG7122B, or AWG7122C
Real Time Oscilloscope	Tektronix DPO/DSA70000, DPO/DSA70000B, or DPO/DSA70000C Series
	Tektronix MSO70000 or MSO70000C Series
	Specified model oscilloscopes with more than 12.5 GHz sampling rate are supported
Error Detector/Analyzer	DPO72004B, DPO72004C, DSA72004B, DSA72004C, DPO71604B, DPO71604C, DSA71604B, DSA71604C, DPO71254B, DPO71254C, DSA71254B, DSA71254C, MSO71254,MSO71254C, MSO71604, MSO71604C, MSO72004, MSO72004C
	Ellisys Protocol Analyzer
Test Fixture	USB-IF test fixture or Tektronix test fixture

Testing a Device Using Normative Test Approach

In the Normative test approach, the frequencies and amplitude considered for analysis are provided in the specifications document. Follow the steps below for testing a device with error detector:

1. Select the **DUT Type** as Device.

2. Enter the **DUT ID**.

3. Select the **Channel Emulation** as Hardware, if using the USB-IF test fixtures and a 3 meter cable.

4. Select the**Test Name** as Normative Receiver Jitter Tolerance Test.
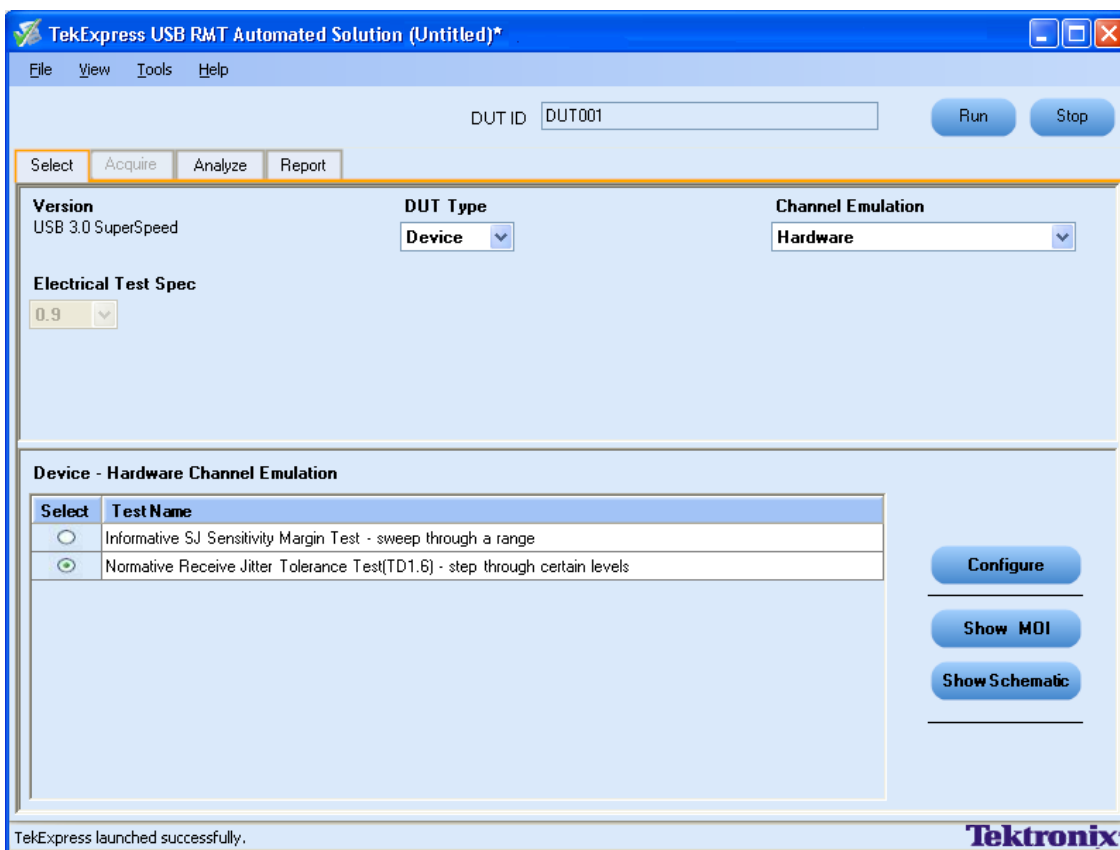
5. If you want to verify the test setup before running the test(s), click **Show Schematic** (see page 99).

6. Click **Configure** to configure the test parameters. Observe that the default settings are in Compliance mode. If you would wish to change the settings to User Defined, you are prompted with a message "If you make changes to a test, the test may no longer be compliant".

7. Select the **Analyze** tab in the configuration screen to include or exclude the various frequencies and amplitude (as specified in the specifications document) for jitter parameters.
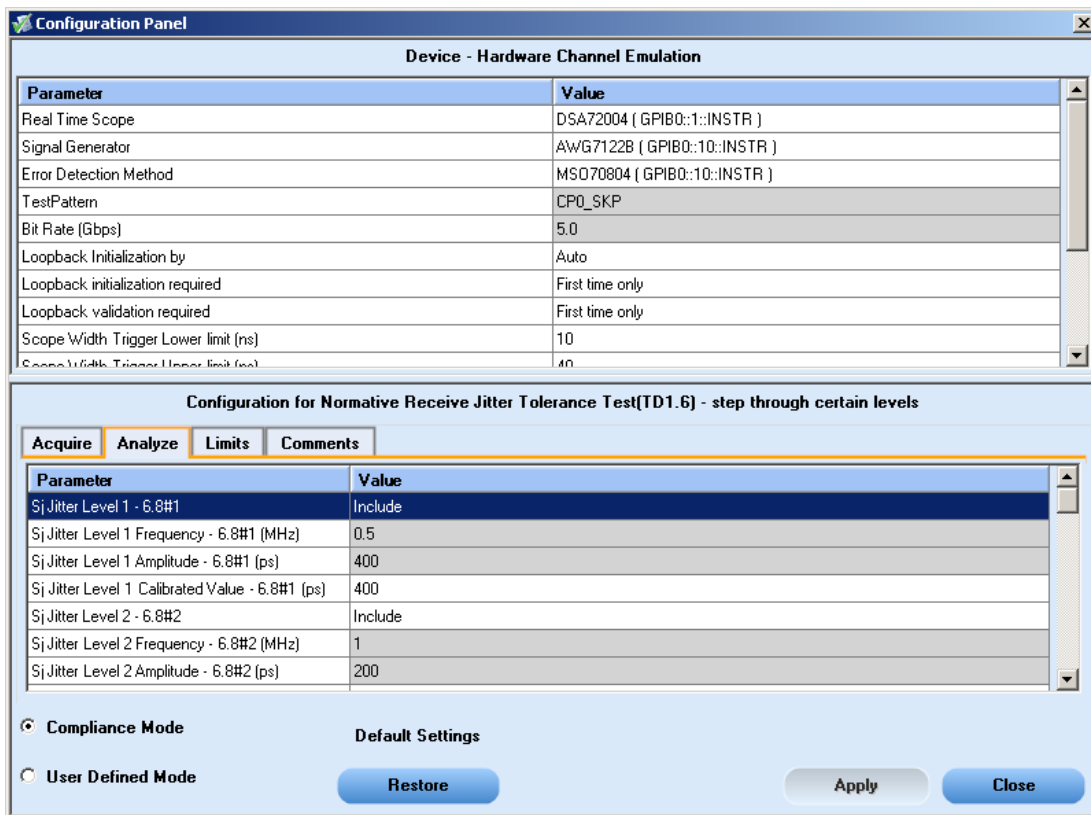
8. Select the **Limits** tab to see the Maximum Allowable limit for error detection. If the number of errors detected by the error detector is more than the specified limit (should be Less Than or Equal To), then the compliance is marked as FAIL.
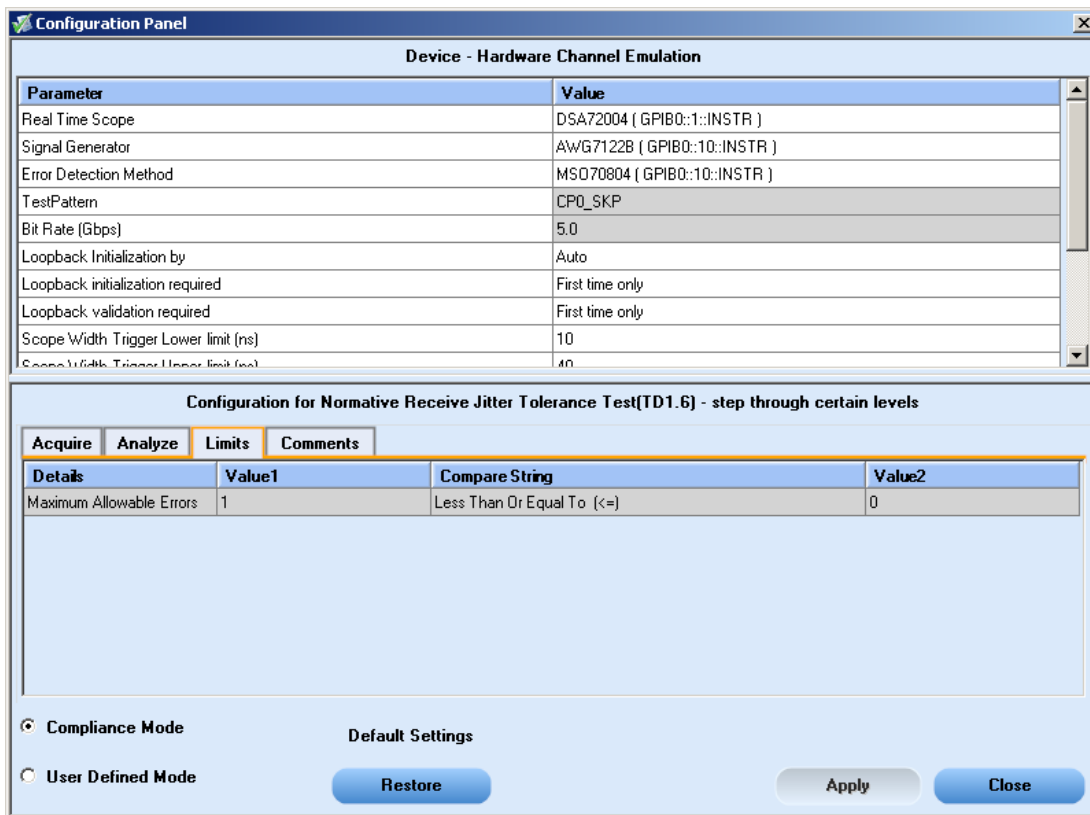
NOTE. You can detect errors using either the oscilloscope error detector or the Ellisys instrument, or manually with other error detectors. If the oscilloscope error detector or the Ellisys instrument is connected, the application automatically counts the number of errors. Otherwise, the application displays a pop-up dialog in which to manually enter the error count.
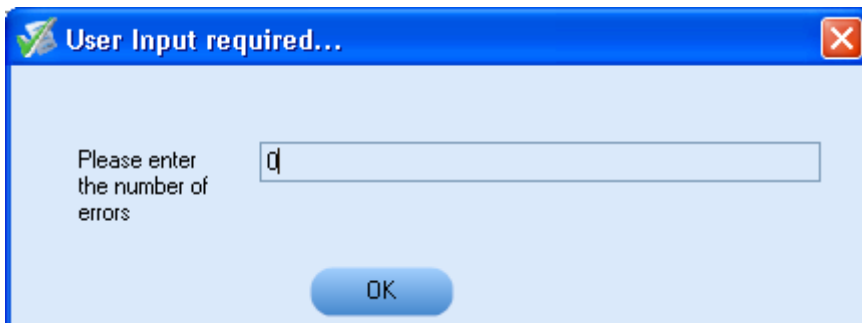
9. Click **Apply** to apply the new settings for the selected test. Click **Close**.

10. Click **Run** to run the selected tests.

The status of the tests is displayed in the Analyze panel, which shows the status of the instruments being initialized, and waveforms being generated for various frequencies. You can also view the graphical results of each test point.
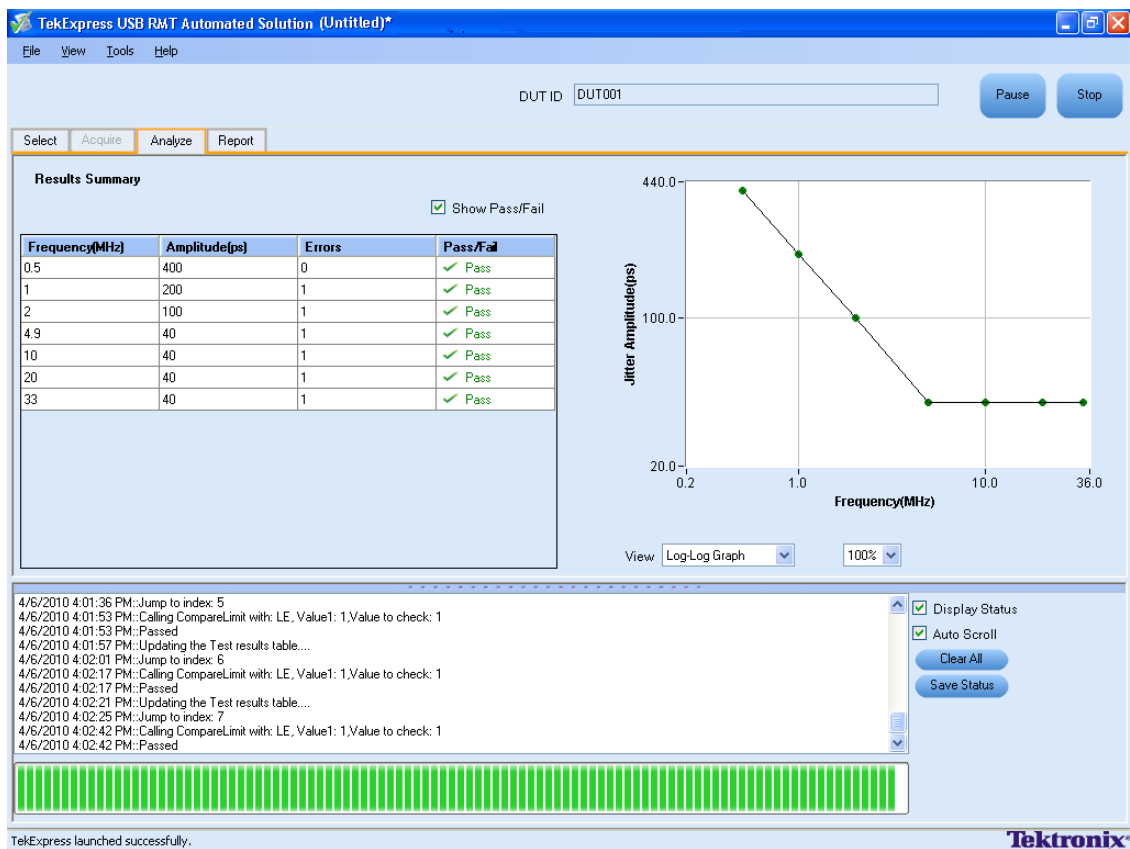
11. The report is automatically updated after each test point and can be viewed in the Reports tab.

> **Ellisys Error Detector**
>
> Ellisys USB Explorer 280 Analyzer/Generator is a USB 3.0 error detector. The hardware is identical to the standard Ellisys Super Speed USB 3.0 protocol analyzer/generator. When used with Tektronix AWG7122B, high-speed signal generator, and automation software from Tektronix, the EX280-CT completes the solution for USB 3.0 receiver testing.

Testing a Device Using Informative Test Approach

In Informative test approach, the frequencies considered are customizable beyond compliance. This approach is used to test the margin of the DUT.

Follow the steps below for testing a device with error detector:

1. Select the **DUT Type** as Device.

2. Enter the **DUT ID**.

3. Select the **Channel Emulation** as Hardware.

4. Select the **Test Name** as Informative SJ Sensitivity Margin Test.

5. If you want to verify the test setup before running the test(s), click Show Schematic (see page 105).

6. Click **Configure** to configure the test parameters. Observe that the default settings are in User Defined mode.
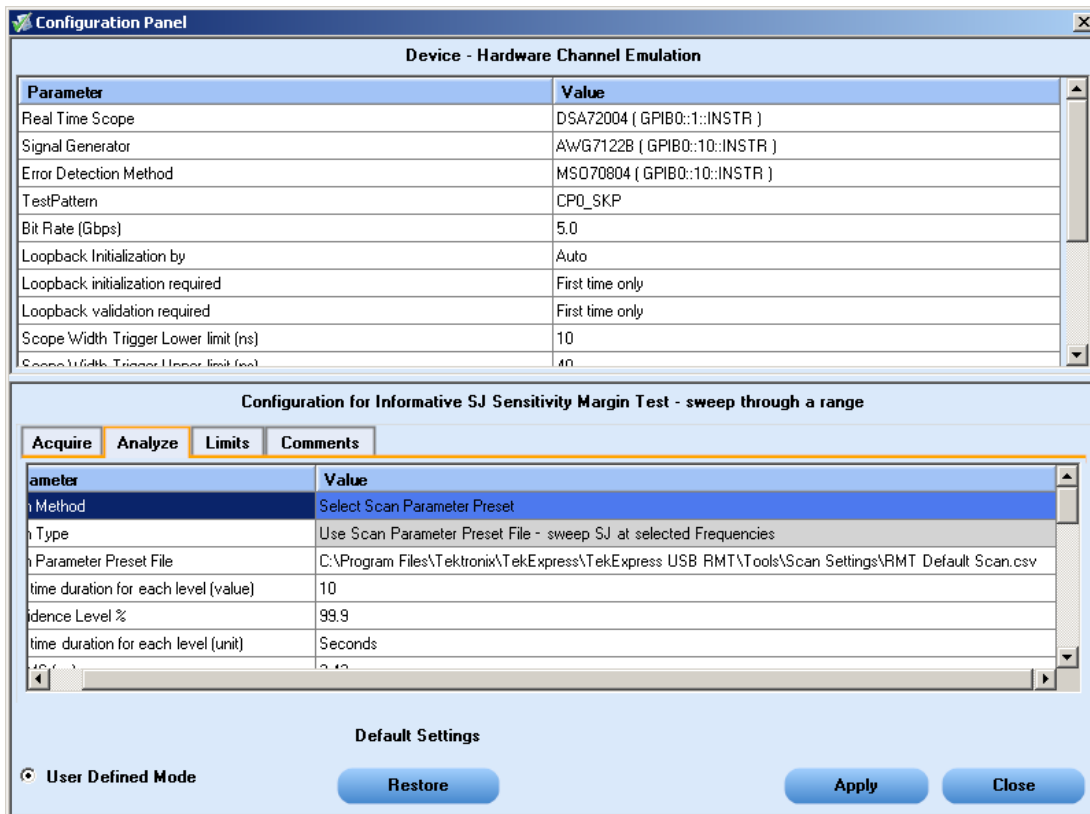
7. Select the **Analyze** tab in the configuration screen and select the following parameters:

Parameter	Description
Scan Method	Select the scan method to locate a failed point: ■ Linear Pass to Fail (see page 41) ■ Select Scan Parameter Preset
Scan Type [1]	The Scan Type parameter is a read-only parameter for information purposes.
Scan Parameter Preset [1]	The path to the parameter preset file.
Start Frequency(MHz) [2]	Specify the SJ start frequency in the MHz.
End Frequency(MHz) [2]	Specify the SJ end frequency in MHz.
Increment Frequency [2]	Specify the incremental frequency in MHz.
Sj Start Value [2]	Specify the jitter start value in ps.

Parameter	Description
Sj End Value [2]	Specify the jitter end value in ps.
Sj increment [2]	Specify the incremental value in ps.

[1] Only shown if the Scan Method is set to **Select Scan Parameter Preset**.

[2] Only shown if the Scan Method is set to **Linear Pass to Fail**.

8. Selecting **Select Scan Parameter Preset** adds the **Scan Parameter Preset File** to the Parameters list. Selecting the Value opens the **Custom Jitter Profile- RMT Default Scan** dialog box. Use this dialog box to enter values for Frequency, Start Sj, Stop Sj, and Step Sj.

- Use the check box in each row to enable or disable tests to execute on the DUT. Tests are executed in the order listed.

- Selecting a test row in the table updates the values in the edit fields. Enter new values in the edit fields and click **Update** to set the values for the selected test row.

- To insert a new test row above the selected row, enter new values in the edit fields and click **Insert**.

- Click **Clear** to clear all the values in the edit fields.

- Click **Save** and enter a file name to save the table settings as a Scan Parameter Preset File to the default location.

- Click **Recall** to recall (load) a saved file or preset.

- Click **Reset** to clear the entire table.

9. Select the **Limits** tab to see the Maximum Allowable limit for error detection. If the number of errors detected by the error detector is more than the specified limit (should be Less Than or Equal To), then the compliance is marked as FAIL.

10. Click **Apply** to apply the new settings for the selected test. Click **Close**.

11. Click **Run** to run the selected tests.

The status of the tests is displayed in the Analyze panel, which shows the pass or fail results of each jitter frequency and amplitude.
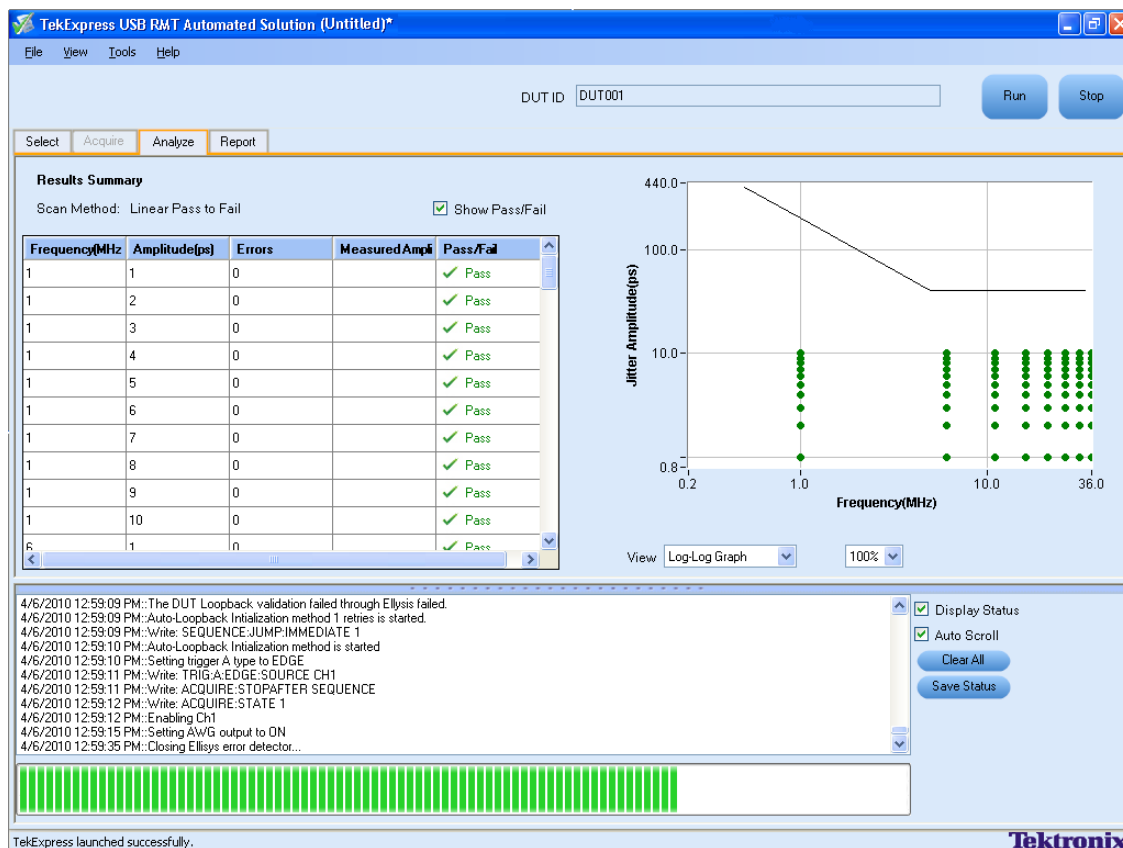
12. The report is automatically updated after each test point and can be viewed in the Reports tab.

Linear Pass to Fail

Uses the specified start amplitude and increases the amount of jitter by the step size until the failure point is located.

About the Programmatic Interface

The Programmatic interface allows you to seamlessly integrate the TekExpress test automation application with the high-level automation layer. This also allows you to control the state of TekExpress application running on a local or a remote PC.

For simplifying the descriptions, the following terminologies are used in this section:

- **TekExpress Client:** A High level automation application that communicates with TekExpress using TekExpress Programmatic Interface.

- **TekExpress Server:** The TekExpress application when being controlled by TekExpress Client.

TekExpress leverages .Net Marshalling to enable the Programmatic Interface for TekExpress Client. TekExpress provides a client library for TekExpress clients to use the programmatic interface. The TekExpress client library is inherited from .Net MarshalByRef class to provide the proxy object for the clients. The TekExpress client library maintains a reference to the TekExpress Server and this reference allows the client to control the server state.
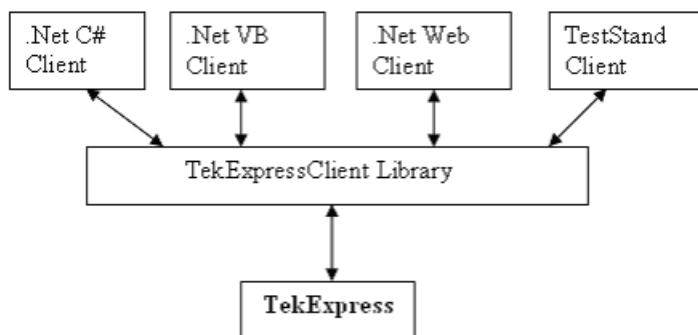
Click the following links to get details on them:

What does one need to have to develop TekExpress Client ?

While developing TekExpress Client one needs to use the TekExpressClient.dll. The client can be a VB .Net, C# .Net, TestStand or web application. The examples for interfaces in each of these applications are in `Samples` folder.

References Required

TekExpressClient.dll has internal reference to *IIdlglib.dll* and *IRemoteInterface.dllIIdlglib.dll* has a reference to*TekDotNetLib.dll*. *IRemoteInterface.dll* provides the interfaces required to perform the remote automations. It is an interface that forms the communication line between the server and the client. *IIdlglib.dll* provides the methods to generate and direct the secondary dialog messages at the client-end.

NOTE. *The end-user client application does not need any reference to above mentioned DLL files. It is essential to have these DLLs (IRemoteInterface.dll, IIdlglib.dll and TekDotNetLib.dll) in same folder location as that of TekExpressClient.dll.*

What steps does a client need to follow ?

The following are the steps that a client needs to follow to use the TekExpressClient.dll to programmatically control the server:

A client UI has to be developed to access the interfaces exposed through the server. This client needs to load `TekExpressClient.dll` to access the interfaces. Once the TekExpressClient.dll is loaded, the client UI can call the specific functions to run the operations requested by the client. Once the client is up and running, it has to do the following to run a remote operation:

1. The client needs to provide the IP address of the PC at which the server is running in order to connect to the server.

2. The client needs to lock the server application to avoid conflict with any other Client that may try to control the server simultaneously. "Lock" would also disable all user controls on server so that server state cannot be changed by manual operation.

3. If any other client tries to access a server which is locked, it will get a notification that the server is locked by another client.

4. When the client has connected to and locked the server, the client can access any of the programmatic controls to run the remote automations.

5. Once the client operations are completed, the server needs to be "unlocked" by the client.

Server and Client Proxy Objects

Remote Proxy Object

The server exposes a remote object to let the remote client access and perform the server side operations remotely. The proxy object is instantiated and exposed at the server-end through marshalling.
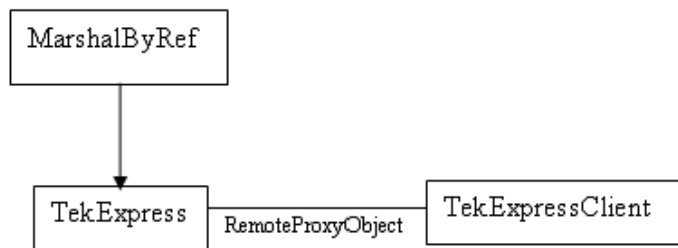
The following is an example:

```
RemotingConfiguration.RegisterWellKnownServiceType (typeof (TekExpressRe-
moteInterface), "TekExpress Remote interface", WellKnownObjectMode.Singleton);
```

This object lets the remote client access the interfaces exposed at the server side. The client gets the reference to this object when the client gets connected to the server.

For example,

//Get a reference to the remote object

```
remoteObject = (IRemoteInterface)Activator.GetObject(typeof(IRemoteInterface),
URL.ToString());
```

Client Proxy Object

Client exposes a proxy object to receive certain information.
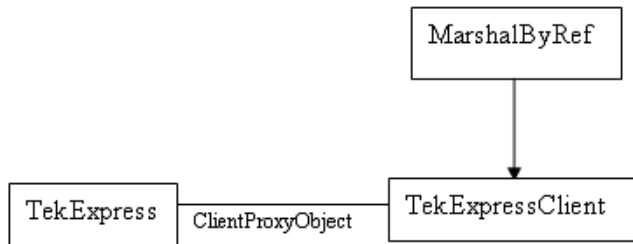
For example,

//Register the client proxy object

```
WellKnownServiceTypeEntry[] e = RemotingConfiguration.GetRegisteredWell-
KnownServiceTypes();
```

```
clientInterface = new ClientInterface();
```

```
RemotingConfiguration.RegisterWellKnownServiceType(typeof(ClientInterface),
"Remote Client Interface", WellKnownObjectMode.Singleton);
```

//Expose the client proxy object through marshalling

```
RemotingServices.Marshal(clientInterface, "Remote Client Interface");
```

The client proxy object is used for the following:

■ To get the secondary dialog messages from the server.

■ To get the file transfer commands from the server while transferring the report.

Click here to see examples.

```
clientObject.clientIntf.DisplayDialog(caption, msg,iconType, btnType);
```

```
clientObject.clientIntf.TransferBytes(buffer, read, fileLength);
```

To know more on the topics below, click the links.

Secondary Dialog Message Handling
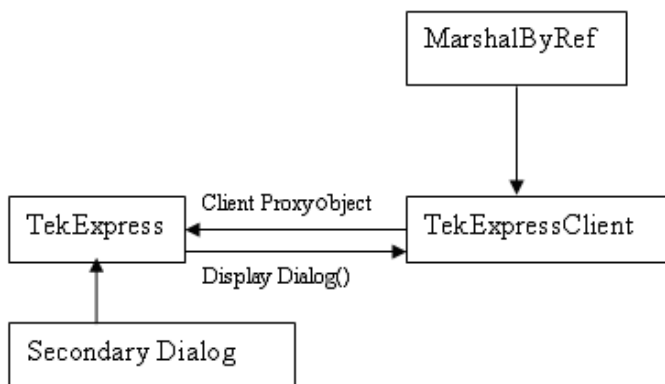
The secondary dialog messages from the Secondary Dialog library are redirected to the client-end when a client is performing the automations at the remote end.

In the secondary dialog library, the assembly that is calling for the dialog box to be displayed is checked and if a remote connection is detected, the messages are directed to the remote end.

File Transfer Events

When the client requests the transfer of the report, the server reads the report and transfers the file by calling the file transfer methods at the client-end.

Client Programmatic Interface: an Example

An example of the client programmatic interface is described and shown as follows:

```
                    ╭─────────────╮
                    │    Start     │
                    ╰──────┬──────╯
                           ▼
              ┌───────────────────────────┐
              │  Connect to Remote Object  │
              └─────────────┬─────────────┘
                           ▼
                 ┌──────────────────┐
                 │   Get Client id   │
                 └────────┬─────────┘
                           ▼
                 ┌──────────────────┐
                 │   Lock Server     │
                 └────────┬─────────┘
                           ▼
        ┌────────────────────────────────────────┐
        │  Register for Status Change Notification │        Handler for Status Change Notification
        └────────────────┬───────────────────────┘
                           ▼                                            ◯
              ┌───────────────────────────┐
              │   Select the requested tests │
              └─────────────┬─────────────┘                           │
                           ▼                                            ▼
  ┌─────────────────────────────────────────────┐              ◇─────────────────◇
  │ Set the necessary Patameters for each selected test │      ◇  Status = "Wait"/ "Error" ◇── No
  └──────────────────────┬──────────────────────┘             ◇─────────────────◇       │
                           ▼                                            │               │
                 ┌──────────────────┐                                   ▼               │
                 │   Run the test    │                       ┌──────────────────┐       │
                 └────────┬─────────┘                        │  Get Status Info   │       │
                           ▼◄──────────────┐                 └────────┬─────────┘       │
                 ┌──────────────────┐       │                         ▼                 │
                 │   Get Status      │       │               ┌────────────────────────┐ │
                 └────────┬─────────┘       │               │ Do Actions based on Status │ │
                           ▼                 │               └───────────┬────────────┘ │
                  ◇──────────────◇           │                           ▼               │
                  ◇ Status= 'Ready' ◇── No ──┘               ┌──────────────────┐       │
                  ◇──────────────◇                           │   Set Response    │       │
                           │                                 └────────┬─────────┘       │
                          Yes                                         ▼◄────────────────┘
                           ▼                      Return to Main Flow  ◯
                 ┌──────────────────┐
                 │   Get Results     │
                 └────────┬─────────┘
                           ▼
              ┌───────────────────────────┐
              │    Process the Results     │
              └─────────────┬─────────────┘
                           ▼
                 ┌──────────────────┐
                 │   Unlock Server   │
                 └────────┬─────────┘
                           ▼
              ┌───────────────────────────┐
              │  Disconnect from Remote Object │
              └─────────────┬─────────────┘
                           ▼
                    ╭─────────────╮
                    │    Stop      │
                    ╰─────────────╯
```

1. Connect to a server or remote object using the programmatic interface provided.

2. Get the client ID that is created when connecting to the remote object. This client ID is one of the required parameters to communicate with the server.

NOTE. *Server identifies the client with this ID only and rejects any request if the ID is invalid.*

3. Lock the server for further operations. This disables the application interface.

NOTE. *You can get values from the server or set values from the server to the client only if application is locked.*

4. Register for receiving notifications on status change events on the server. To register you need to give a handler as a parameter.

NOTE. *Whenever there is a change in the status of the server, all the clients registered with the server receive a notification from the server.*

5. Select the tests that you want to run through the programmatic interface.

6. Set the necessary parameters for each test.

7. Run the tests.

8. Poll for the status of the application.

NOTE. *You can skip this step if you are registered for the status change notification and when the status is Ready.*

9. After completing the tests, get the results.

10. Create a report or display the results and verify or process the results.

11. Unlock the server once you complete all the tasks.

12. Disconnect from the remote object.

Handler of Status Change Notification

1. Get the status. If the status is Wait or Error, get the information which contains the title, message description, and the expected responses for the status.

2. Perform the actions based on the status information.

3. Set the response as expected.

USB-RMT Application Command Arguments and Queries

string id			
Name	**Type**	**Direction**	**Description**
id	string	IN	Identifier of the client that is performing the remote function.

Ready: Test configured and ready to start.

Running: Test running.

Paused: Test paused.

Wait: A popup that needs your inputs.

Error: An error is occurred.

string dutName

Name	Type	Direction	Description
dutName	string	IN	The new DUT ID of the setup.

out bool saved

Name	Type	Direction	Description
saved	bool	OUT	Boolean representing whether the current session is saved.

This parameter is used as a check in SaveSession() and SaveSessionAs() functions.

string ipAddress

Name	Type	Direction	Description
ipAdress	string	IN	The ip address of the server to which the client is trying to connect. This is required to establish the connection between the server and the client.

out string clientID

Name	Type	Direction	Description
clientid	String	OUT	Identifier of the client that is connected to the server. clientId = unique number + ipaddress of the client. For example, 1065–192.157.98.70

NOTE. *If the dutName parameter is null, the client is prompted to provide a valid DUT ID.*

NOTE. *The server must be active and running for the client to connect to the server. Any number of clients can be connected to the server at a time.*

NOTE. *When the client is disconnected, it is unlocked from the server and then disconnected. The id is reused.*

string dutId

Name	Type	Direction	Description
dutId	string	OUT	The DUT ID of the setup.

The dutId parameter is set after the server processes the request.

string device

Name	Type	Direction	Description
device	string	IN	Specifies the name of the device.

string suite

Name	Type	Direction	Description
suite	string	IN	Specifies the name of the suite.

string test

Name	Type	Direction	Description
test	string	IN	Specifies the name of the test from which to obtain the pass or fail status.

string parameterString

Name	Type	Direction	Description
parameterString	string	IN	Selects or deselects a test.

int rowNr

Name	Type	Direction	Description
rowNr	int	IN	Specifies the zero based row index of the sub-measurement for obtaining the result value.

NOTE. *When the client tries to lock a server that is locked by another client, the client gets a notification that the server is already locked and it must wait until the server is unlocked. If the client locks the server and is idle for a certain amount of time then the server is unlocked automatically from that client.*

out string[] status

Name	Type	Direction	Description
status	string array	OUT	The list of status messages generated during run.

string name

Name	Type	Direction	Description
name	string	IN	The name of the session being recalled.

The name parameter cannot be empty. If it is empty, the client is prompted to provide a valid name.

NOTE. *When the run is performed, the status of the run is updated periodically using a timer.*

---

string name

Name	Type	Direction	Description
name	string	IN	The name of the session being saved.

The name parameter cannot be empty. If it is empty, the client is prompted to provide a valid name.

Once the session is saved under 'name' you cannot use this method to save the session in a different name. Use SaveSessionAs instead.

---

string name

Name	Type	Direction	Description
name	string	IN	The name of the session being recalled.

The same session is saved under different names using this method. The name parameter cannot be empty. If it is empty, the client is prompted to provide a valid name.

---

bool isSelected

Name	Type	Direction	Description
isSelected	bool	IN	Selects or deselects a test.

string time

Name	Type	Direction	Description
time	string	IN	The time in seconds which refers to the timeout period.

The time parameter gives the timeout period, that is the time the client is allowed to be locked and idle. After the timeout period if the client is still idle, it gets unlocked.

The time parameter should be a positive integer. Else, the client is prompted to provide a valid timeout period.

bool_verbose

Name	Type	Direction	Description
_verbose	bool	IN	Specifies whether thec verbose mode should be turned ON or OFF.

NOTE. *When the session is stopped, the client is prompted to stop the session and is stopped at the consent.*

string filePath

Name	Type	Direction	Description
filePath	string	IN	The location where the report must be saved in the client.

NOTE. *If the client does not provide the location to save the report, the report is saved at C:\ProgramFiles.*

NOTE. *When the client is disconnected, the client is automatically unlocked.*

out string caption

Name	Type	Direction	Description
caption	String	OUT	The wait state or error state message sent to you.

out string message

Name	Type	Direction	Description
message	String	OUT	The wait state /error state message to you.

out string[] buttonTexts

Name	Type	Direction	Description
buttonTexts	string array	OUT	An array of strings containing the possible response types that you can send.

string response

Name	Type	Direction	Description
response	string	IN	A string containing the response type that you can select (it must be one of the strings in the string array buttonTexts).

out string clientID

Name	Type	Direction	Description
clientID	String	OUT	Identifier of the client that is connected to the server.
			clientID = unique number + ipaddress of the client. For example, 1065–192.157.98.70

Connect Through an IP Address

Table 9: Connect through an IP address

Command name	Parameters	Description	Return Value	Example

Connect()	string ipAddress (see page 51)
	
	out string clientID (see page 51)
		This method connects the client to the server.
		Note (see page 51)
		The client provides the IP address to connect to the server.
		The server provides a unique client identification number when connected to it.
			True or False.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL
				returnval as boolean
				returnval = m_Client.Con-nect(ipaddress,m_clientID)

Table 10: Test- Connect through an IP address

Parameters	Description	Return Value	Example
Command name: Connect()			

string ipAddress (see page 51)
out string clientID (see page 51)
	This method connects the client to the server.
	Note (see page 51)
	The client provides the IP address to connect to the server.
	The server provides a unique client identification number when connected to it.
		True or False.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL
			returnval as boolean
			returnval = m_Client.Con-nect(ipaddress,m_clientID)

NOTE. The Fail condition for PI commands occurs in any of the following cases:

The server is LOCKED and the message displayed is "Server is locked by another client".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

Lock the Server

Table 11: Lock the server

Command name	Parameters	Description	Return Value	Example
LockSession()	string clientID (see page 55)	This method locks the server. Note (see page 52) The client must call this method before running any of the remote automations. The server can be locked by only one client.	String value that gives the status of the operation after it has been performed. The return value is "Session Locked..." on success.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string returnval = m_Client.LockServer(clientID)

NOTE. The Fail condition for PI commands occurs in any of the following cases:

The server is LOCKED and the message displayed is "Lock Session to execute the command".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

Disable the Popups

Table 12: Disable the popups

Command name	Parameters	Description	Return Value	Example
SetVerboseMode()	string clientID (see page 55) bool _verbose (see page 54)	This method sets the verbose mode to either true or false. When the value is set to true, then any of the message boxes appearing during the application are routed to the client machine which is controlling TekExpress. When the value is set to false, then all the message boxes are shown on the server machine.	String that gives the status of the operation after it has been performed. When Verbose mode is set to true, the return value is "Verbose mode turned on. All dialog boxes are shown to client ...". When Verbose mode is set to false, the return value is "Verbose mode turned off. All dialog box will be shown to server ...".	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string **Verbose mode is turned on** return=m_Client.SetVerbose-Mode(clientID, true) **Verbose mode is turned off** returnval=m_Client.SetVerbose-Mode(clientID, false)

NOTE. *The Fail condition for PI commands occurs in any of the following cases:*

The server is LOCKED and the message displayed is "Lock Session to execute the command".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

Set or Get the DUT ID

Table 13: Set or Get the DUT ID

Command name	Parameters	Description	Return Value	Example
SetDutId()	string clientID (see page 55) string dutName (see page 51)	This method changes the DUT ID of the set up. The client must provide a valid DUT ID.	String that gives the status of the operation after it has been performed. Return value is "DUT Id Changed..." on success.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string return=m_Client.SetDutId(clientID,desiredDutId) Note (see page 51)
GetDutId()	string clientID (see page 55) string dutId (see page 52)	This method gets the DUT ID of the current set up.	String that gives the status of the operation after it has been performed.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string return=m_Client.GetDutid(clientID, out DutId)

NOTE. The Fail condition for PI commands occurs in any of the following cases:

The server is LOCKED and the message displayed is "Lock Session to execute the command".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

Set the Configuration Parameters for a Suite or Measurement

Table 14: Set the configuration parameters for a suite or measurement

Command name	Parameters	Description	Return Value	Example
SetGeneralParameter()	string clientID (see page 55) string device (see page 52) string suite (see page 52) string test (see page 52) string parameterString (see page 52)	This method sets the general parameters that are not specific to any given suite or measurement. *NOTE. Using this command we can select a lane, channel, or source type.*	String that gives the status of the operation after it has been performed. The return value is "" (an empty String) on success.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string Select Channel (see page 60) Select Source Type (Differential) (see page 61) Loopback Initialization (see page 61) Width Trigger Lower Limit (see page 61)
SetAnalyzeParameter()	string clientID (see page 55) string device (see page 52) string suite (see page 52) string test (see page 52) string parameterString (see page 52)	This method sets the configuration parameters in the Analyze panel of the Configuration Panel dialog box for a given suite or measurement.	The return value is "" (an empty String) on success.	m_Client = new Client() //m_Clientis a reference to the Client class in the Client DLL returnval as string

NOTE. The Fail condition for PI commands occurs in any of the following cases:

The server is LOCKED and the message displayed is "Lock Session to execute the command".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

Select Channel Example

returnval=mClient.SetGeneralParameter(clientID, "Device", "Suite", "Lane AConnected to$Channel 1")

Select Source Type (Differential) Example

returnval=mClient.SetGeneralParameter(clientID, "Device", "Suite", "Source Type$Differential")

LoopBack Initialization Example

returnval=mClient.SetGeneralParameter(clientID, "Device", "Receiver", "Loopback Initialization by$Auto")

Width Trigger Lower Limit Example

returnval=mClient.SetGeneralParameter(clientID, "Device", "Receiver", "Scope Width Trigger Lower limit (ns)$40")

Query the Configuration Parameters for a Suite or Measurement

Table 15: Query the configuration parameters for a suite or measurement

Command name	Parameters	Description	Return Value	Example
GetGeneralParameter()	string clientID (see page 55) string device (see page 52) string suite (see page 52) string test (see page 52) string parameterString (see page 52)	This method gets the general configuration parameters for a given suite or measurement.	The return value is the general configuration parameter for a given suite or measurement that is set.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string
GetAnalyzeParameter()	string clientID (see page 55) string device (see page 52) string suite (see page 52) string test (see page 52) string parameterString (see page 52)	This method gets the configuration parameters set in the Analyze panel of the Configuration Panel dialog box for a given suite or measurement.	The return value is the configuration parameter set in the Analyze panel of the Configuration Panel dialog box for a given suite or measurement.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string

NOTE. The Fail condition for PI commands occurs in any of the following cases:

The server is LOCKED and the message displayed is "Lock Session to execute the command".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

Select a Measurement

Table 16: Select a measurement

Command name	Parameters	Description	Return Value	Example
SelectTest()	string clientID (see page 55) string device (see page 52) string suite (see page 52) string test (see page 52) bool isSelected (see page 53)	This method selects or deselects a given test. Setting parameter isSelected to true, you can select a measurement. Setting parameter isSelected to false, you can deselect a measurement.	String that displays the status of the operation after it has been performed. The return value is "" (an empty String) on success.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string returnval=mClient.Se-lectTest(clientID, "Device", "Suite", " Linearity->Tone-2", True) returnval=mClient.SelectTest(clientID, "Device", "Receiver", "Normative Receive Jitter Tolerance Test(TD1.6) - step through certain levels", True)

NOTE. The Fail condition for PI commands occurs in any of the following cases:

The server is LOCKED and the message displayed is "Lock Session to execute the command".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

Select a Suite

Table 17: Select a suite

Command name	Parameters	Description	Return Value	Example
SelectSuite()	string clientID (see page 55) string device (see page 52) string suite (see page 52) bool isSelected (see page 53)	This method selects or deselects a given suite. Setting parameter isSelected to true, you can select a suite. Setting parameter isSelected to false, you can deselect a suite.	String that gives the status of the operation after it has been performed. The return value is "" (an empty String) on success.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string **Select Suite (Default):** returnval=m_Client.Select-Suite(clientID, "Device", "Suite", true)

NOTE. *The Fail condition for PI commands occurs in any of the following cases:*

The server is LOCKED and the message displayed is "Lock Session to execute the command".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

Run with Set Configurations or Stop the Run Operation

Table 18: Run with set configurations or Stop the run operation

Command name	Parameters	Description	Return Value	Example
Run()	string clientID (see page 55)	Runs the selected measurements. Note (see page 53) Once the server is set up and is configured, it can be run remotely using this function.	String that gives the status of the operation after it has been performed. The return value is "Run started..." on success.	m_Client = new Client() //m_Clientis a reference to the Client class in the Client DLL returnval as string returnval=m_Client.Run(clientID)
Stop()	string clientID (see page 55)	Stops the currently running measurements. Note (see page 54)	String that gives the status of the operation after it has been performed. The return value is "Stopped..." on success.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string returnval=m_Client.Stop(clientID)

NOTE. *The Fail condition for PI commands occurs in any of the following cases:*

The server is LOCKED and the message displayed is "Lock Session to execute the command".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

Get or Set the Timeout Value

Table 19: Get or Set the Timeout value

Command name	Parameters	Description	Return Value	Example
GetTimeOut()	string clientID (see page 55)	Returns the current timeout period set by the client.	String that gives the status of the operation after it has been performed. The default return value is 1800000.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string returnval=m_Client.GetTime-Out()
SetTimeOut()	string clientID (see page 55) string time (see page 54)	Sets a timeout period specified by client. After expiry of this timeout period, the server is automatically unlocked.	String that gives the status of the operation after it has been performed. On success the return value is "TimeOut Period Changed".	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string returnval=m_Client.SetTime-Out(clientID, desiredTimeOut)

NOTE. *The Fail condition for PI commands occurs in any of the following cases:*

The server is LOCKED and the message displayed is "Lock Session to execute the command".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

Wait for the Test to Complete

The commands in this group are executed while tests are running. The GetCurrentStateInfo() and SendResponse() commands are executed when application is running and in wait state.

Table 20: Wait for the test to complete

Command name	Parameters	Description	Return Value	Example
ApplicationStatus()	string clientID (see page 55)	This method gets the status of the server application. The states at a given time are Ready, Running, Paused, Wait, or Error. (see page 50)	String value that gives the status of the server application.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string returnval=m_Client.Applica-tionStatus(clientID)

QueryStatus()	string clientID (see page 55)
	
	out string[] status (see page 53)
		It is an interface for the user to transfer Analyze panel status messages from the server to the client.	String that gives the status of the operation after it has been performed.
			
			On success the return value is "Transferred...".
				m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL
				
				returnval as string
				
				returnVal=m_Client.QueryS-tatus(clientID, out statusMes-sages)
				
				if ((OP_STATUS)returnVal == OP_STATUS.SUCCESS)
				
				return "Status updated..."
				
				else
				
				return CommandFailed(re-turnVal)

Table 20: Wait for the test to complete (cont.)

Command name	Parameters	Description	Return Value	Example
GetCurrentState-Info() **NOTE.** *This command is used when the application is running and is in the wait or error state.*	string clientID (see page 55) out string caption (see page 54) out string message (see page 55) out string[] buttonTexts (see page 55)	This method gets the additional information of the states when the application is in Wait or Error state. Except client ID, all the others are out parameters.	This command does not return any value. This function fills up the out parameters that are passed when invoking this function.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL m_Client.GetCurrentState-Info()
SendResponse() **NOTE.** *This command is used when the application is running and is in the wait or error state.*	string clientID (see page 55) out string caption (see page 54) out string message (see page 55) string response (see page 55)	After receiving the additional information using the method GetCurrentState-Info(), the client can decide on the response to send and send the response to the application using this function. The response should be one of the strings that was earlier received as a string array in the GetCurrentState-Info function. The _caption and _message should match the information received earlier in the GetCurrentStateInfo function.	This command does not return any value.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL m_Client.SendResponse()

NOTE. *The Fail condition for PI commands occurs in any of the following cases:*

The server is LOCKED and the message displayed is "Lock Session to execute the command".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

After the Test is Complete

Table 21: After the test is complete

Command name	Parameters	Description	Return Value	Example

GetPassFailStatus()	string clientID (see page 55)
	
	string device (see page 52)
	
	string suite (see page 52)
	
	string test (see page 52)
		This method gets the pass or fail status of the measurement after test completion.
		
		NOTE. *Execute this command after completing the measurement.*
			String that gives the status of the operation after it has been performed.
			
			Returns the pass or fail status in the form of a string.
				m_Client = new Client()
				//m_Client is a reference to the Client class in the Client DLL
				
				returnval as string
				
				returnval=m_Client.GetPass-FailStatus(clientID, device, suite) //Pass or Fail

GetResultsValue()	string clientID (see page 55) string device (see page 52) string suite (see page 52) string test (see page 52) string parameterString (see page 52)	This method gets the result values of the measurement after the run.	String that gives the status of the operation after it has been performed. Returns the result value in the form of a string.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as srting
GetResultsValue-ForSubMeasurements()	string clientID (see page 55) string device (see page 52) string suite (see page 52) string test (see page 52) string parameterString (see page 69) int rowNr (see page 52)	This method gets the result values for individual sub-measurements, after the run.	String that gives the status of the operation after it has been performed. Returns the result value in the form of a string.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string

Table 21: After the test is complete (cont.)

Command name	Parameters	Description	Return Value	Example
TransferReport()	string clientID (see page 55) string filePath (see page 54)	This method transfers the report generated after the run. The report contains the summary of the run. The client must provide the location where the report is to be saved at the client-end.	String that gives the status of the operation after it has been performed. Transfers all the result values in the form of a string.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string returnval=m_Client.TransferReport(clientID,"C:\Report")
TransferWave-forms()	string clientID (see page 55) string filePath (see page 54)	This method transfers all the waveforms from the folder for the current run. **NOTE.** *For each click of Run button, a folder is created in the X: drive. Transfer the waveforms before clicking the Run button.*	String that gives the status of the operation after it has been performed. Transfers all the waveforms in the form of a string. On success the return value is "Transferred...".	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string returnval=m_Client.Transfer-Waveforms(clientID,"C:\Wave-forms")

NOTE. *The Fail condition for PI commands occurs in any of the following cases:*

The server is LOCKED and the message displayed is "Lock Session to execute the command".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

string parameterString

Name	Type	Direction	Description
parameterString	string	IN	Specifies the oscilloscope model, TekExpress version, and USB-RMT version.

Save, Recall, or Check if a Session is Saved

Table 22: Save, Recall, or Check if a session is saved

Command name	Parameters	Description	Return Value	Example
CheckSession-Saved()	string clientID (see page 55) out bool saved (see page 51)	This method is called when a check is to be made to know if the current session is saved.	Return value is either True or False.	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string returnval=m_Client.Check-SessionSaved(m_clientID, out savedStatus)
RecallSession()	string clientID (see page 55) string name (see page 53)	Recalls a saved session. The name of the session is provided by the client.	String that gives the status of the operation after it has been performed. The return value is "Session Recalled...".	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string returnval=m_Client.RecallSes-sion(clientID, savedSession-Name)
SaveSession()	string clientID (see page 55) string name (see page 53)	Saves the current session. The name of the session is provided by the client.	String that gives the status of the operation after it has been performed. The return value is "Session Saved..."/"Failed...".	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string returnval=m_Client.SaveSes-sion(clientID, desiredSession-Name)
SaveSessionAs()	string clientID (see page 55) string name (see page 53)	Saves the current session in a different name every time this method is called. The name of the session is provided by the client.	String that gives the status of the operation after it has been performed. The return value is "Session Saved...".	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string returnval=m_Client.SaveSes-sionAs(clientID, desiredSes-sionName)

NOTE. *The Fail condition for PI commands occurs in any of the following cases:*

The server is LOCKED and the message displayed is "Lock Session to execute the command".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

Unlock the Server

Table 23: Unlock the server

Command name	Parameters	Description	Return Value	Example
UnlockSession()	string clientID (see page 55)	This method unlocks the server from the client. The ID of the client to be unlocked must be provided. Note (see page 54)	String that gives the status of the operation after it has been performed. The return value is "Session Un-Locked...".	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string returnval=m_Client.Unlock-Server(clientID)

NOTE. *The Fail condition for PI commands occurs in any of the following cases:*

The server is LOCKED and the message displayed is "Lock Session to execute the command".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

Disconnect from the Server

Table 24: Disconnect from the server

Command name	Parameters	Description	Return Value	Example
Disconnect()	string clientID (see page 55)	This method disconnects the client from the server it is connected to. Note (see page 51)	Integer value that gives the status of the operation after it has been performed. 1 for Success –1 for Failure	m_Client = new Client() //m_Client is a reference to the Client class in the Client DLL returnval as string returnval=m_Client.Discon-nect(m_clientID)

NOTE. *The Fail condition for PI commands occurs in any of the following cases:*

The server is LOCKED and the message displayed is "Lock Session to execute the command".

The session is UNLOCKED and the message displayed is "Lock Session to execute the command".

The server is NOTFOUND and the message displayed is "Server not found...Disconnect!".

When none of these fail conditions occur, then the message displayed is "Failed...".

Set or Get Configuration Parameters for Normative Receiver Test

Lists the configuration parameters in the Analyze panel of the configuration dialog box for Normative Receiver Test (Compliance):

Select and Configure Test Frequency and Amplitude

Table 25: Configure test frequency and amplitude parameters

	Parameters	Default Value	Options/Range	Example
Select and configure test frequency & Jitter #1	Sj Jitter Level 1 - 6.8#1	Include	Include, Exclude	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 1 - 6.8#1$Exclude")
				String str =GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 1 - 6.8#1)"
	Sj Jitter Level 1 Frequency - 6.8#1 (MHz)	0.5	0.1–100	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 1 Frequency - 6.8#1 (MHz)$50")
				String str =GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 1 Frequency - 6.8#1 (MHz)")
	Sj Jitter Level 1 Amplitude - 6.8#1 (ps)	400	0–1000000	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 1 Amplitude - 6.8#1 (ps)$100")
				String str =GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 1 Amplitude - 6.8#1 (ps)")

Table 25: Configure test frequency and amplitude parameters (cont.)

	Parameters	Default Value	Options/Range	Example
Select and configure test frequency & Jitter #2	Sj Jitter Level 2 - 6.8#2	Include	Include, Exclude	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 2 - 6.8#2$Exclude") String str =GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 2 - 6.8#2)
	Sj Jitter Level 2 Frequency - 6.8#2 (MHz)	1	1–100	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 2 Frequency - 6.8#2 (MHz)$50") String str =GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 2 Frequency - 6.8#2 (MHz)")
	Sj Jitter Level 2 Amplitude - 6.8#2 (ps)	200	0–1000000	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 2 Amplitude - 6.8#2 (ps)$100") String str =GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 2 Amplitude - 6.8#2 (ps)")

Table 25: Configure test frequency and amplitude parameters (cont.)

	Parameters	Default Value	Options/Range	Example
Select and configure test frequency & Jitter #3	Sj Jitter Level 3 - 6.8#3	Include	Include, Exclude	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 3 - 6.8#3$Exclude")
				String str =GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 3 - 6.8#3)
	Sj Jitter Level 3 Frequency - 6.8#3 (MHz)	2	0.1–100	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 3 Frequency - 6.8#3 (MHz)$50")
				String str =GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 3 Frequency - 6.8#3 (MHz)")
	Sj Jitter Level 3 Amplitude - 6.8#3 (ps)	200	0–1000000	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 3 Amplitude - 6.8#3 (ps)$100")
				String str =GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 3 Amplitude - 6.8#3 (ps))

Table 25: Configure test frequency and amplitude parameters (cont.)

	Parameters	Default Value	Options/Range	Example
Select and configure test frequency & Jitter #4	Sj Jitter Level 4 - 6.8#4	Include	Include,Exclude	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 4 - 6.8#4$Exclude")
				String str =GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 4 - 6.8#4")
	Sj Jitter Level 4 Frequency - 6.8#4 (MHz)	4.9	0.1–100	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 4 Frequency - 6.8#4 (MHz)$50")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 4 Frequency - 6.8#4 (MHz)")
	Sj Jitter Level 4 Amplitude - 6.8#4 (ps)	40	0–1000000	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 4 Amplitude - 6.8#4 (ps)$100")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 4 Amplitude - 6.8#4 (ps)")

Table 25: Configure test frequency and amplitude parameters (cont.)

	Parameters	Default Value	Options/Range	Example
Select and configure test frequency & Jitter #5	Sj Jitter Level 5 - 6.8#5	Include	Include,Exclude	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 5 - 6.8#5$Exclude") String str =GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 5 - 6.8#5")
	Sj Jitter Level 5 Frequency - 6.8#5 (MHz)	10	0.1–100	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 5 Frequency - 6.8#5 (MHz)$50") String strGetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 5 Frequency - 6.8#5 (MHz)")
	Sj Jitter Level 5 Amplitude - 6.8#5 (ps)	40	0–1000000	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 5 Amplitude - 6.8#5 (ps)$100") String strGetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 5 Amplitude - 6.8#5 (ps)")

Table 25: Configure test frequency and amplitude parameters (cont.)

	Parameters	Default Value	Options/Range	Example
Select and configure test frequency & Jitter #6	Sj Jitter Level 6 - 6.8#6	Include	Include,Exclude	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 6 - 6.8#6$Exclude")
				String str =GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 6 - 6.8#6")
	Sj Jitter Level 6 Frequency - 6.8#6 (MHz)	20	0.1–100	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 6 Frequency - 6.8#6 (MHz)$50")
				String str=GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 6 Frequency - 6.8#6 (MHz)")
	Sj Jitter Level 6 Amplitude - 6.8#6 (ps)	40	0–1000000	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 6 Amplitude - 6.8#6 (ps)$100")
				String str=GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 6 Amplitude - 6.8#6 (ps)")

Table 25: Configure test frequency and amplitude parameters (cont.)

	Parameters	Default Value	Options/Range	Example
Select and configure test frequency & Jitter #7	Sj Jitter Level 7 - 6.8#7	Include	Include,Exclude	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 7 - 6.8#5$Exclude")
				String str =GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 7 - 6.8#7")
	Sj Jitter Level 7 Frequency - 6.8#7 (MHz)	33	0.1–100	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 7 Frequency - 6.8#7 (MHz)$50")
				String str=GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 7 Frequency - 6.8#7 (MHz)")
	Sj Jitter Level 7 Amplitude - 6.8#7 (ps)	40	0–1000000	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 7 Amplitude - 6.8#7 (ps)$100")
				String str=GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Sj Jitter Level 7 Amplitude - 6.8#7 (ps)")

Test Time and System Configuration for Normative Test

Table 26: Test time and system configuration for normative test parameters

	Parameters	Value	Options/Range	Example
Test Time Configuration	Test time duration for each level (value)	10	0 - 1000	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Test time duration for each level (value)$20")
				SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Test time duration for each level (value)")
	Confidence Level %	99.9	50–99.999	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Confidence Level %$99.999")
				string str= GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Confidence Level")
	Test time duration for each level (unit)	Seconds	BER rate 10 to the -exponent, seconds	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Test time duration for each level (unit))$BER rate 10 to the -exponent")
				String str: GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Test time duration for each level (unit))")
Test System Configuration	Waveform Library Location	C:\US-BRMT	String - File path should be entered	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Waveform Library Location$D:\USB-RMT New")
				String str=GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Waveform Library Location)
	Waveform Creation Option	Always Create Waveform	Always Create Waveform; Use Pre-created Waveforms; Use pre-created waveforms and create if not available	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Waveform Creation Option$Use Pre-created Waveforms")
				String str=GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Waveform Creation Option")
				Frequency and jitter configuration parameters
	Save Created Waveform	Always	Always, Never	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Save Created Waveform$Never")
				String str=GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Save Created Waveform")

Waveform Creation for Normative Test

Table 27: Waveform creation parameters for normative test

	Parameters	Default	Options-/Range	Example
Waveform Creation Configuration	Rj RMS (ps)	2.42	0 - 10000	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Rj RMS (ps)$20")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Rj RMS (ps)")
	Tj (ps)	7.5	0 - 1000000	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Tj (ps)$20")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Tj (ps)")
	SSC	With SSC	With SSC, Without SSC	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "SSC$Without SSC")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "SSC")
	SSC Shape	Triangle	Sine, Triangle	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "SSC Shape$Sine")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "SSC Shape")
	SSC Spread	Down	Down, Up, Center, Unequal	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "SSC Spread$Up")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "SSC Spread")

	Parameters	Default	Options-/Range	Example
Waveform Creation Configuration	Frequency Deviation (ppm)	5000	0 - 200000	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Frequency Deviation (ppm)$5500")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Frequency Deviation (ppm)")
	Frequency Modulation (KHz)	33	10 - 500	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Frequency Modulation (KHz)$200")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Frequency Modulation (KHz)")
	Pre-emphasis (dB)	3	0 - 20	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Pre-emphasis (dB)$10")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Pre-emphasis (dB)")
	Rise Time (ps)	30	30 - 120	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Rise Time (ps)$60")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Rise Time (ps)")
	Fall Time (ps)	30	30 - 120	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Fall Time (ps)$50")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Fall Time (ps)")

	Parameters	Default	Options-/Range	Example
Waveform Creation Configuration	Amplitude Minimum	–0.4	–0.5 – 0.5	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Amplitude Minimum$0.3")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Amplitude Minimum")
	Amplitude Maximum	0.4	–0.5 - 0.5	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Amplitude Maximum$0.3")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Amplitude Maximum")
	Touchstone 4-Port Data Type	Single-ended	Single-ended, Differential	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "Touchstone 4-Port Data Type$Differential")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "Touchstone 4-Port Data Type")
	S-Parameter file 1	C:\USBRMT\USB3_Front_Cable_Device_CTLE 2.s4p	Browse the filepath	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "S-Parameter file 1$C:\New File.s4p")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "S-Parameter file 1")
	S-Parameter file 2	C:\USBRMT\USB3_Front_Cable_Device_CTLE 1.s4p	Browse the filepath	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "S-Parameter file 2$C:\New File.s4p")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "S-Parameter file 2")
Waveform Creation Configuration	S-Parameter file 3	C:\USBRMT\USB3_Front_Cable_Device_CTLE 3.s4p	Browse the filepath	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "S-Parameter file 3$C:\New File.s4p")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "S-Parameter file 3")
	S-Parameter file 4	C:\USBRMT\USB3_Front_Cable_Device_CTLE 4.s4p	Browse the filepath	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "S-Parameter file 4$C:\New File.s4p")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "S-Parameter file 4")
	S-Parameter file 5	C:\USBRMT\USB3_Front_Cable_Device_CTLE 5.s4p	Browse the filepath	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "S-Parameter file 5$C:\New File.s4p")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "S-Parameter file 5")
	S-Parameter file 6	C:\USBRMT\USB3_Front_Cable_Device_CTLE 6.s4p	Browse the filepath	SetAnalyzeParameter(clientId, device, suite, normativeTestName, "S-Parameter file 6$C:\New File.s4p")
				GetAnalyzeParameter(clientId, device, suite, normativeTestName, "S-Parameter file 6")

Set or Get Configuration Parameters for Informative Test

Frequency and Jitter Configuration

Table 28: Frequency and jitter configuration parameters

	Parameters	Value	Options/Range	Example
Frequency configuration	Start Frequency (MHz)	1	0.1 – 1000	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Start Frequency (MHz)$20")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Start Frequency (MHz)")
	End Frequency (MHz)	60	0.1 – 1000	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "End Frequency (MHz)$20")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "End Frequency (MHz)")
	Increment Frequency (MHz)	5	1 – 100	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Increment Frequency (MHz)$20")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Increment Frequency (MHz)$2")

	Parameters	Value	Options/Range	Example
Jitter Con-figuration	Sj Start Value (ps)	1	0 - 1000000	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Sj Start Value (ps)$20")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Sj Start Value (ps)")
	Sj End Value (ps)	10	0 - 1000000	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Sj End Value (ps)$20")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Sj End Value (ps)")
	Sj Increment (ps)	1	0 - 1000	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Sj Increment (ps)$200")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Sj Increment (ps)")
Scanning method	Scan Method	Linear Pass to Fail	Linear Pass to Fail IOptions: Linear Pass to Fail Select Scan Parameter Preset	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Scan Method$Linear Pass to Fail")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Scan Method")
	Scan Parameter Preset File	C:\Program Files\Tek-tronix\TekEx-press\TekEx-press USB RMT\Tools\Scan Settings\RMT Default Scan.csv	Browse the file path	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Scan Pa-rameter Preset File$C:\Program Files\Tek-tronix\TekExpress\TekExpress USB RMT\Tools\Scan Settings\RMT Default Scan.csv") GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Scan Method")

Test Time and System Configuration for Informative Test

Table 29: Test time and system configuration for informative test parameters

	Parameters	Value	Options/Range	Example
Test Time Configuration	Test time duration for each level (value)	10	0 - 1000	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Test time duration for each level (value)$20")
				SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Test time duration for each level (value)")
	Confidence Level %	99.9	50–99.999	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Confidence Level %$99.999")
				string str= GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Confidence Level")
	Test time duration for each level (unit)	Seconds	BER rate 10 to the -exponent, seconds	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Test time duration for each level (unit))$BER rate 10 to the -exponent")
				String str: GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Test time duration for each level (unit))")
Test System Configuration	Waveform Library Location	C:\U-SB-RMT	String - File path should be entered	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Waveform Library Location$D:\USB-RMT New")
				String str=GetAnalyzeParameter(clientId, device, suite, informativeTestName,"Waveform Library Location)
	Waveform Creation Option	Always Create Waveform	Always Create Waveform, Use Pre-created Waveforms, Use pre-created waveforms and create if not available	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Waveform Creation Option$Use Pre-created Waveforms")
				String str=GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Waveform Creation Option")
	Save Created Waveform	Always	Always, Never, Save Last Pass and Last Fail Waveforms	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Save Created Waveform$Never")
				String str=GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Save Created Waveform")

Waveform Creation Configuration for Informative Test

Table 30: Waveform creation configuration for informative test parameters

	Parameters	Default	Options-/Range	Example
Waveform Creation Configuration	Rj RMS (ps)	2.42	0 - 10000	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Rj RMS (ps)$20")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Rj RMS (ps)")
	Tj (ps)	7.5	0 - 1000000	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Tj (ps)$20")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Tj (ps)")
	SSC	With SSC	With SSC, Without SSC	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "SSC$Without SSC")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "SSC")
	SSC Shape	Triangle	Sine, Triangle	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "SSC Shape$Sine")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "SSC Shape")
	SSC Spread	Down	Down, Up, Center, Unequal	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "SSC Spread$Up")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "SSC Spread")
	Frequency Deviation (ppm)	5000	0 - 200000	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Frequency Deviation (ppm)$5500")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Frequency Deviation (ppm)")
	Frequency Modulation (KHz)	33	10 - 500	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Frequency Modulation (KHz)$200")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, , "Frequency Modulation (KHz)")

Table 30: Waveform creation configuration for informative test parameters (cont.)

	Parameters	Default	Options-/Range	Example
Waveform Creation Configuration	Pre-emphasis (dB)	3	0 - 20	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Pre-emphasis (dB)$10")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Pre-emphasis (dB)")
	Rise Time (ps)	30	30 - 120	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Rise Time (ps)$60")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Rise Time (ps)")
	Fall Time (ps)	30	30 - 120	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Fall Time (ps)$50")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Fall Time (ps)")
	Amplitude Minimum	–0.4	–0.5 – 0.5	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Amplitude Minimum$0.3")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Amplitude Minimum")
	Amplitude Maximum	0.4	–0.5 - 0.5	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Amplitude Maximum$0.3")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Amplitude Maximum")
	Touchstone 4-Port Data Type	Single-ended	Single-ended, Differential	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "Touchstone 4-Port Data Type$Differential")
				GetAnalyzeParameter(clientId, device, suite, informativeTestName, "Touchstone 4-Port Data Type")

Table 30: Waveform creation configuration for informative test parameters (cont.)

	Parameters	Default	Options-/Range	Example
Waveform Creation Configuration	S-Parameter file 1	C:\US-BRMT\USB-3_Front_Cable_Device_CTLE 2.s4p	Browse the filepath	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "S-Parameter file 1$C:\New File.s4p") GetAnalyzeParameter(clientId, device, suite, informativeTestName, "S-Parameter file 1")
	S-Parameter file 2	C:\US-BRMT\USB-3_Front_Cable_Device_CTLE 1.s4p	Browse the filepath	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "S-Parameter file 2$C:\New File.s4p") GetAnalyzeParameter(clientId, device, suite, informativeTestName, "S-Parameter file 2")
	S-Parameter file 3	C:\US-BRMT\USB-3_Front_Cable_Device_CTLE 3.s4p	Browse the filepath	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "S-Parameter file 3$C:\New File.s4p") GetAnalyzeParameter(clientId, device, suite, informativeTestName, "S-Parameter file 3")
	S-Parameter file 4	C:\US-BRMT\USB-3_Front_Cable_Device_CTLE 4.s4p	Browse the filepath	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "S-Parameter file 4$C:\New File.s4p") GetAnalyzeParameter(clientId, device, suite, informativeTestName, "S-Parameter file 4")
	S-Parameter file 5	C:\US-BRMT\USB-3_Front_Cable_Device_CTLE 5.s4p	Browse the filepath	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "S-Parameter file 5$C:\New File.s4p") GetAnalyzeParameter(clientId, device, suite, informativeTestName, "S-Parameter file 5")
	S-Parameter file 6	C:\US-BRMT\USB-3_Front_Cable_Device_CTLE 6.s4p	Browse the filepath	SetAnalyzeParameter(clientId, device, suite, informativeTestName, "S-Parameter file 6$C:\New File.s4p") GetAnalyzeParameter(clientId, device, suite, informativeTestName, "S-Parameter file 6")

General Parameters

Table 31: General parameters

	Parameters	Default	Options/Range	Example
Instru-ment Config-uration	Real Time Scope	\<Instru-ment Ad-dress\>	List from Instrument discovery	
	Signal Generator	\<Instru-ment Ad-dress\>	List from Instrument discovery	
	Error Detection Method	\<Instru-ment Ad-dress\>	List from Instrument discovery	
Other Param-eters	TestPattern	CP0_SKP	CP0, MinAdd1P, BERC, BRST, MinAdd1N, CP0_SKP, CP1, CP2, CP3, CP4, CP6, CP8	SetGeneralParameter(clientID, device, suite, normativeTestName, "TestPattern$CP0") GetGeneralParameter(clientID, device, suite, normativeTestName, "TestPattern")
	Bit Rate (Gbps)	5	1–10	SetGeneralParameter(clientID, device, suite, normativeTestName, "Bit Rate (Gbps)$3") GetGeneralParameter(clientID, device, suite, normativeTestName, "Bit Rate (Gbps)")
	Loopback Initialization by	Auto	Auto, Custom Utility, User Defined Batch Script	SetGeneralParameter(clientID, device, suite, normativeTestName, "Loopback Initialization by$User Defined Batch Script") GetGeneralParameter(clientID, device, suite, normativeTestName, "Loopback Initialization by")
	Loopback initialization required	First time only	Always, First time only	SetGeneralParameter(clientID, device, suite, normativeTestName, "Loopback initialization required$Always") GetGeneralParameter(clientID, device, suite, normativeTestName, "Loopback initialization required")

	Parameters	Default	Options/Range	Example
Other Param- eters	Loopback validation required	First time only	Always, First time only,Never	SetGeneralParameter(clientID, device, suite, normativeTestName, "Loopback validation required$Always")
				GetGeneralParameter(clientID, device, suite, normativeTestName, "Loopback validation required")
	Measure Jitter Method	Do Not Mea- sure	Using Scope, Manual, Do Not Measure	SetGeneralParameter(clientID, device, suite, normativeTestName, "Measure Jitter Method$Using Scope")
				GetGeneralParameter(clientID, device, suite, normativeTestName, "Measure Jitter Method")
	Channel Emulation	Hard- ware	Hardware, Software	SetGeneralParameter(clientID, device, suite, normativeTestName, "Channel Emulation$Software")
				GetGeneralParameter(clientID, device, suite, normativeTestName, "Channel Emulation")
	Number of retries for instrument IO errors	3	0–5	SetGeneralParameter(clientID, device, suite, normativeTestName, "Number of retries for instrument IO errors$5")
				GetGeneralParameter(clientID, device, suite, normativeTestName, "Number of retries for instrument IO errors")
	Number of retries for Auto Loopback initialization	3	0–5	SetGeneralParameter(clientID, device, suite, normativeTestName, "Number of retries for Auto Loopback initialization$5")
				GetGeneralParameter(clientID, device, suite, normativeTestName, "Number of retries for Auto Loopback initialization")
	Time between retries (seconds)	20	5–60	SetGeneralParameter(clientID, device, suite, normativeTestName, "Time between retries (seconds)$25")
				GetGeneralParameter(clientID, device, suite, normativeTestName, "Time between retries (seconds)")

Handle Error Codes

The return value of the remote automations at the server-end is OP_STATUS which is changed to a string value depending on its code and returned to the client. The values of OP_STATUS are as follows:

Value	Code	Description
FAIL	−1	The operation failed.
SUCCESS	1	The operation succeeded.
NOTFOUND	2	Server not found.
LOCKED	3	The server is locked by another client, so operation cannot be performed.
UNLOCK	4	The server is not locked. Lock the server before performing the operation.
NULL	0	Nothing.

Program Example

This is a reference program to illustrate how to communicate to TekExpress USB-RMT remotely.

A typical application does the following:

1. Start the application.

2. Connect through an IP address.

   `m_Client.Connect("localhost")` 'True or False

   `clientID = m_Client.getClientID`

3. Lock the server.

   `m_Client.LockServer(clientID)`

4. Disable the Popups.

   `m_Client.SetVerboseMode(clientID, false)`

5. Set the Dut ID.

   `m_Client.SetDutId(clientID, "DUT_001")`

6. Select Informative SJ Sensitivity Margin Test - sweep through a range.

   `m_Client.SelectSingleTest(clientID, "Device", "Receiver", "No Version", " Normative Receive Jitter Tolerance Test(TD1.6) - step through certain levels ")` 'Receive Jitter Tolerance Test(TD1.6) - step through certain levels selected

7. Select channel emulation.

```
mClient.SetGeneralParameter(clientID, "Device", "Receiver", "Channel
Emulation$Hardware") 'Channel Emulation set to Hardware
```

8. Configure the selected measurement.

```
mClient.SetAnalyzeParameter(clientID, "Device", "Suite", "Sj Jitter Level 2
- 6.8#2$Exclude") "Sj Jitter Level 2 – 6.8#2 excluded from test
```

```
mClient.SetAnalyzeParameter(clientID, "Device", "Suite", "Sj Jitter Level 2
- 6.8#3$Exclude") "Sj Jitter Level 2 – 6.8#3 excluded from test
```

```
mClient.SetAnalyzeParameter(clientID, "Device", "Suite", "Sj Jitter Level 2
- 6.8#5$Exclude") "Sj Jitter Level 2 – 6.8#5 excluded from test
```

```
mClient.SetAnalyzeParameter(clientID, "Device", "Suite", "Sj Jitter Level 2
- 6.8#6$Exclude") "Sj Jitter Level 2 – 6.8#6 excluded from test
```

9. Run with set configurations.

```
m_Client.Run(clientID)
```

10. Wait for the test to complete.

```
Do

Thread.Sleep(500)

m_Client.Application_Status(clientID)

Select Case status

Case "Wait"
```

'Get the Current State Information

```
mClient.GetCurrentStateInfo(clientID, WaitingMsbBxCaption, WaitingMsbBxMes-
sage, WaitingMsbBxButtontexts)
```

'Send the Response

```
mClient.SendResponse(clientID, WaitingMsbBxCaption, WaitingMsbBxMessage,
WaitingMsbBxResponse)

End Select

Loop Until status = "Ready"
```

11. After the Test is Complete.

'Save all results values from folder for current run

```
m_Client.TransferResult(clientID, logDirname)
```

'Save all images from folder for current run

```
m_Client.TransferImages(clientID, logDirname)
```

12. Unlock the server.

```
m_Client.UnlockServer(clientID)
```

13. Disconnect from server.

```
m_Client.Disconnect()
```

14. Exit the application.

NI TestStand Client Example

The following is an example for NI TestStand Client available in the path, `C:\Program Files\Tektronix\TekExpress\TekExpress USB-RMT\Examples\Programmatic Interface\TestStand\TekExpressClient_USBRMT.seq, ComplianceTestExample sequence`
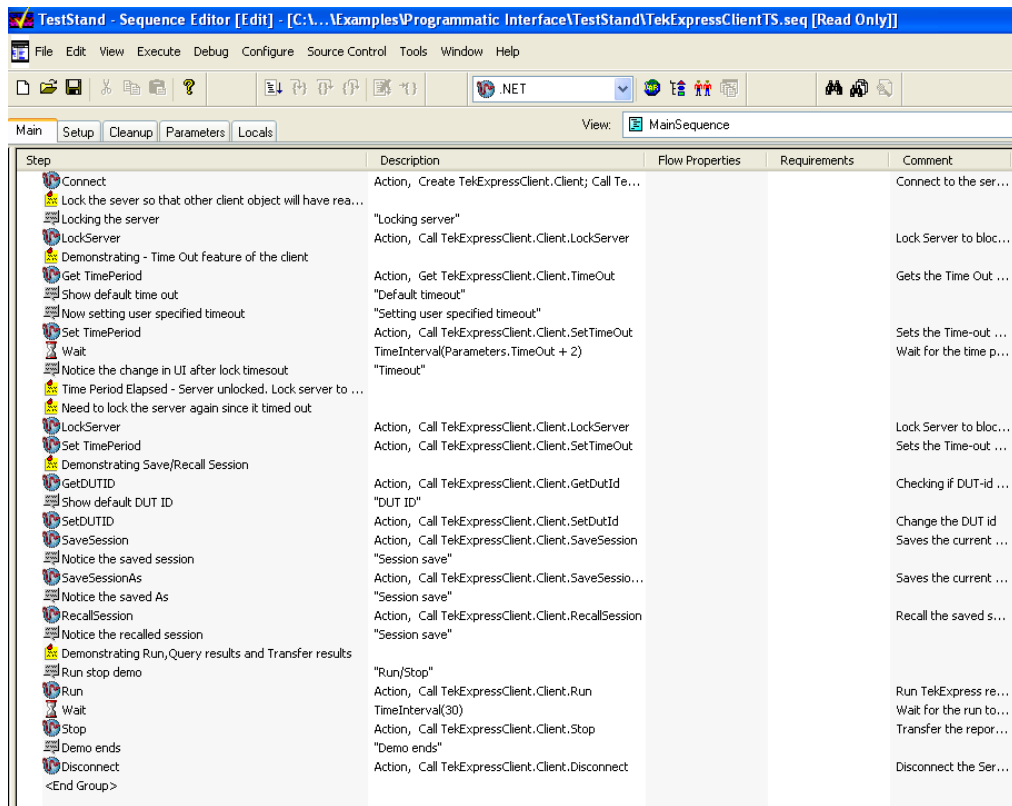
Example

The following is an example for NI TestStand Client available in the path, `C:\Program Files\Tektronix\TekExpress\Examples\Programmatic Interface\TestStand`.
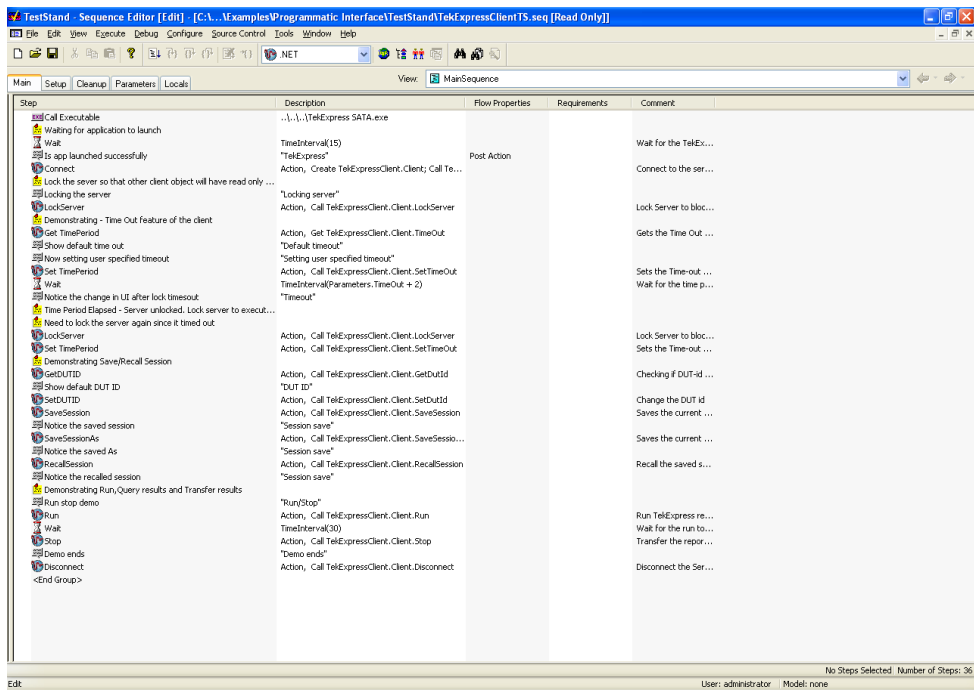
Instrument Connectivity

If the instrument(s) are displayed in TekVISA Instrument Manager but not in the TekExpress Instrument Bench, check the following:

- Only those instruments that respond to `*idn?` and `*opt?` queries successfully, are displayed in Instrument Bench.

- Ensure that VXI-11 Server is running on the instruments.

If Instrument initialization fails during test sequence execution, do the following:

It is observed that GPIB communication with instrument over Tek-VISA layer is not initialized if in TekVISA Instrument manager the search criteria is turned-off even if valid instrument is connected in the network. It is necessary to turn ON the respective search criteria by opening the TekVISA Instrument manager.

NOTE. Ellisys must be connected using Non-VISA. An error is displayed when you run the test without the signal generator being detected.

TestStand Runtime Engine Installation

Managing multiple versions of TestStand installed on the system.

TekExpress installs TestStand version 4.2.1 runtime engine. If you have other versions of TestStand installed as well, make sure that TestStand version 4.2.1 is active while working with TekExpress. To set the active version of TestStand used by TekExpress, click **Start > Programs > National Instruments > TestStand 4.2.1 > TestStand Version Selector**.
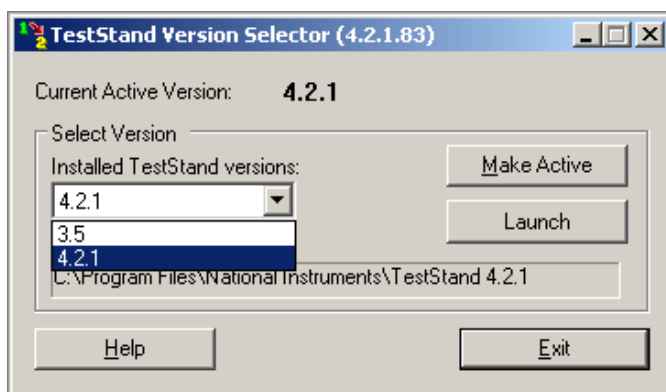
Instrument Connectivity using GPIB Cable

The following schematic illustrates the GPIB connectivity of instruments when TekExpress is running on a stand-alone computer.

Instrument connectivity - GPIB Power Supply
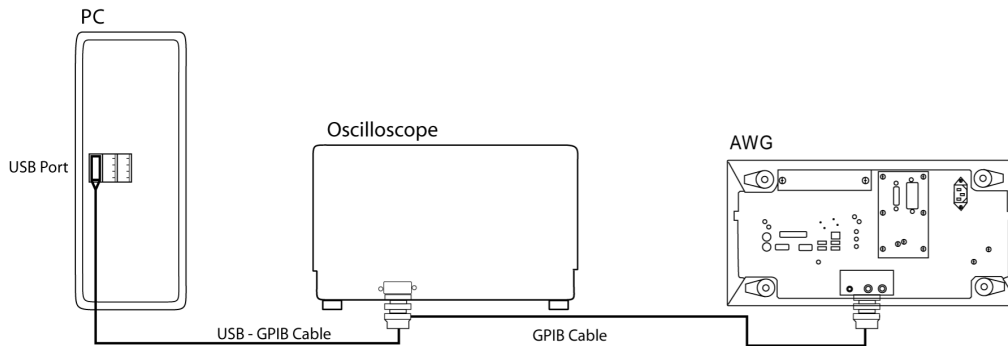
The following schematic illustrates the GPIB connectivity of instrument when TekExpress is running on an oscilloscope.

Instrument connectivity - GPIB Power Supply
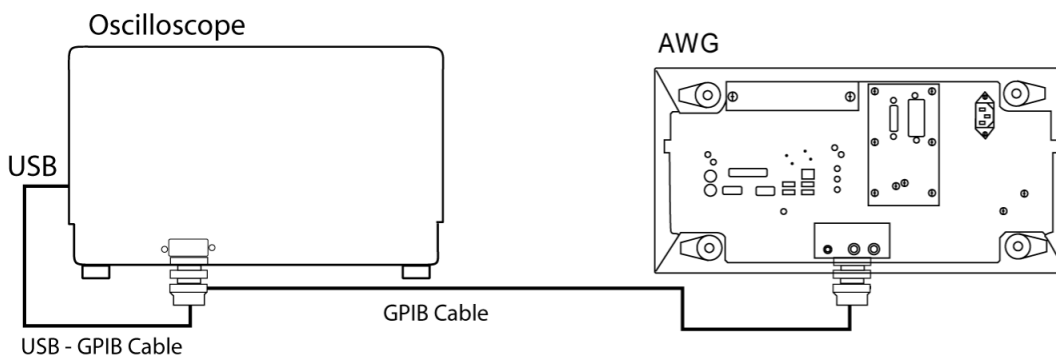
Schematics for Normative Test Approach

The following setup is required to test a device or a host using Normative/Informative Test approach:

Device-Software Channel Emulation-Ellisys Error Detector
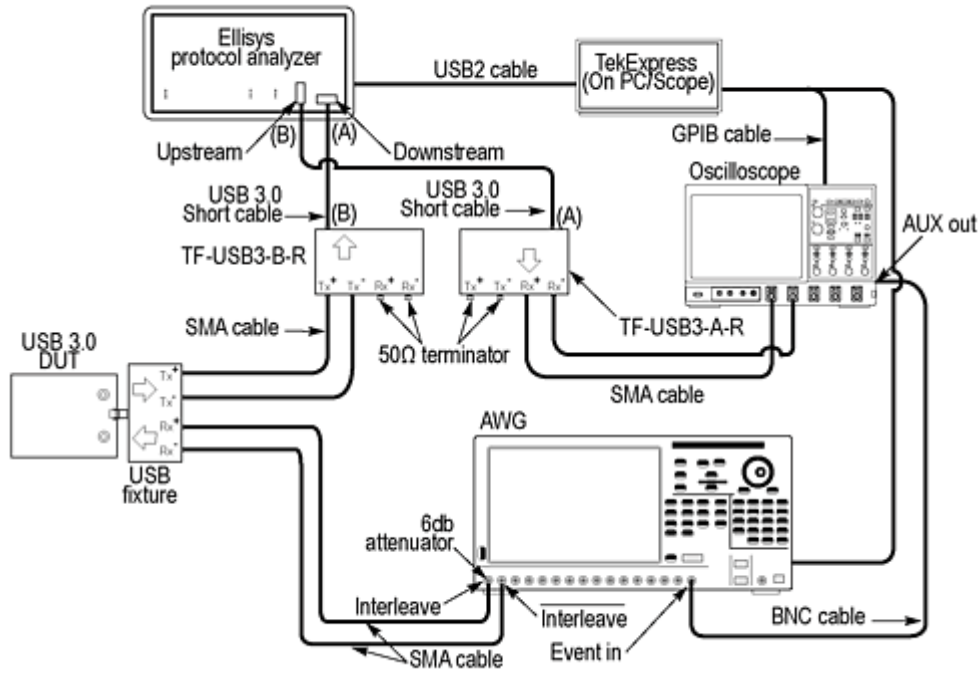
Device-Hardware Channel Emulation-Ellisys Error Detector

Device-Hardware Channel Emulation-Scope Error Detector

Host-Hardware Channel Emulation-Scope Error Detector

Host-Software Channel Emulation-Ellisys Error Detector

Host-Hardware Channel Emulation-Ellisys Error Detector

Host-Software Channel Emulation-Scope Error Detector

Host-Hardware Channel Emulation-Scope Error Detector

Oscilloscope

AUX out

BNC cable

GPIB cable

SMA cable

USB 3.0 DUT

USB3.0 cable

Tx⁺

Tx⁻

AWG

USB
fixture

6db
attenuator

USB 3.0 cable

Interleave

Interleave

Rx⁺

Rx⁻

SMA cable

Event in

USB fixture
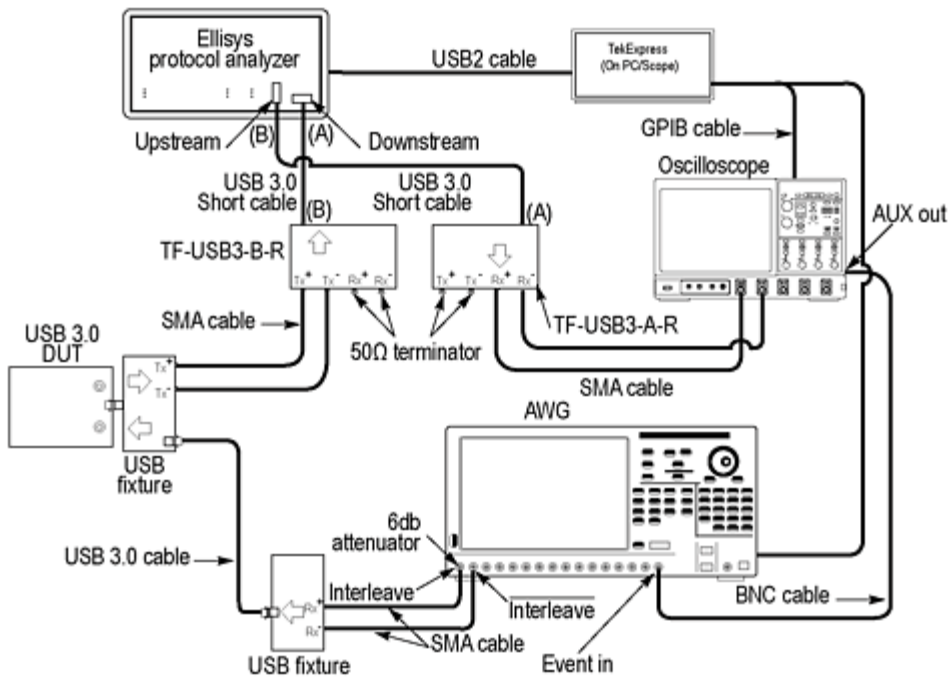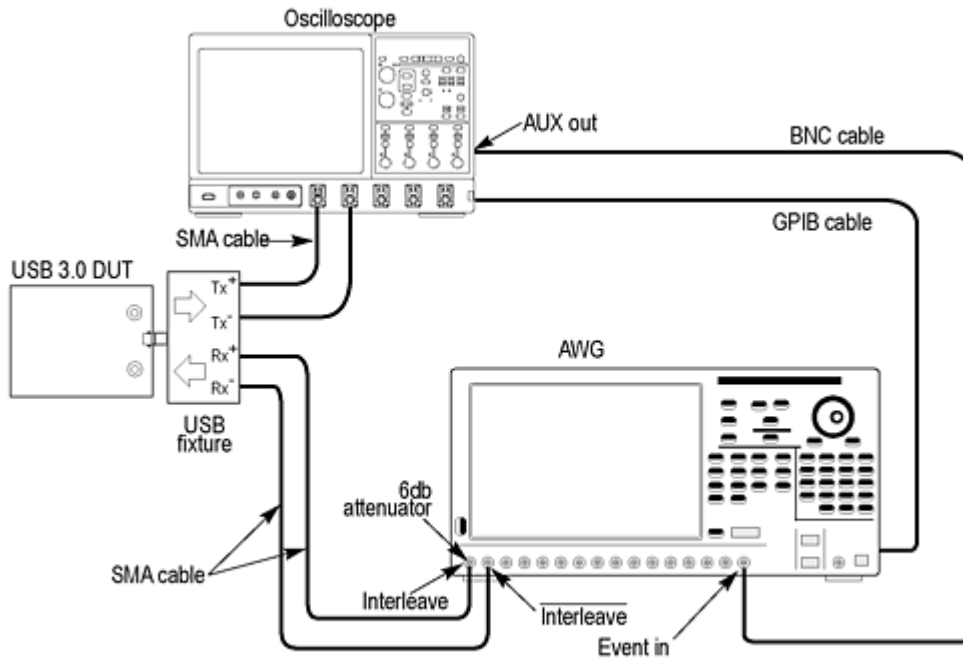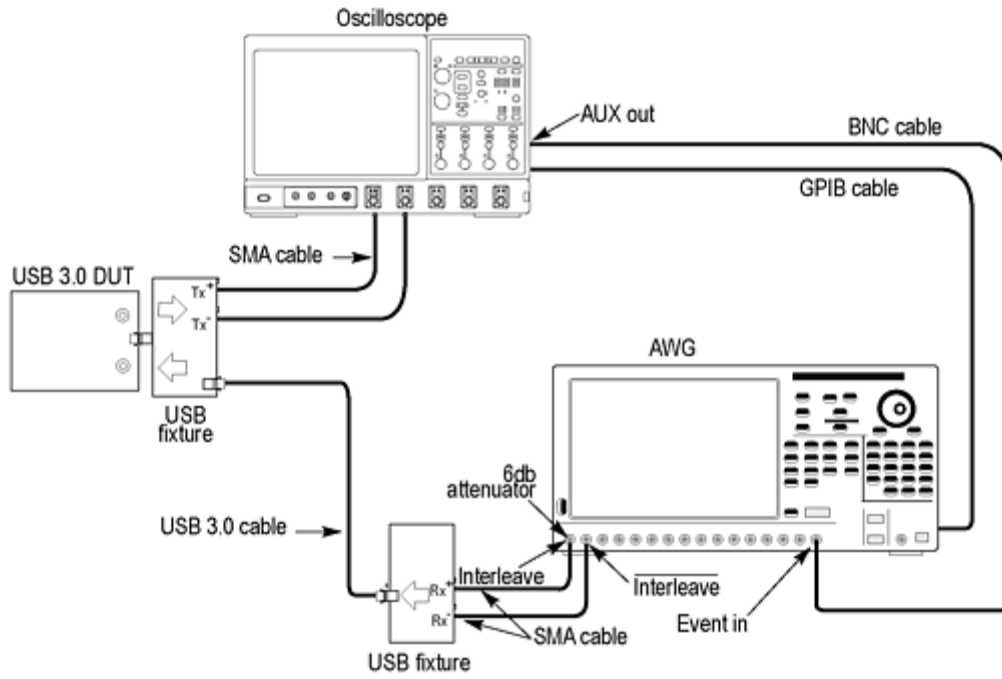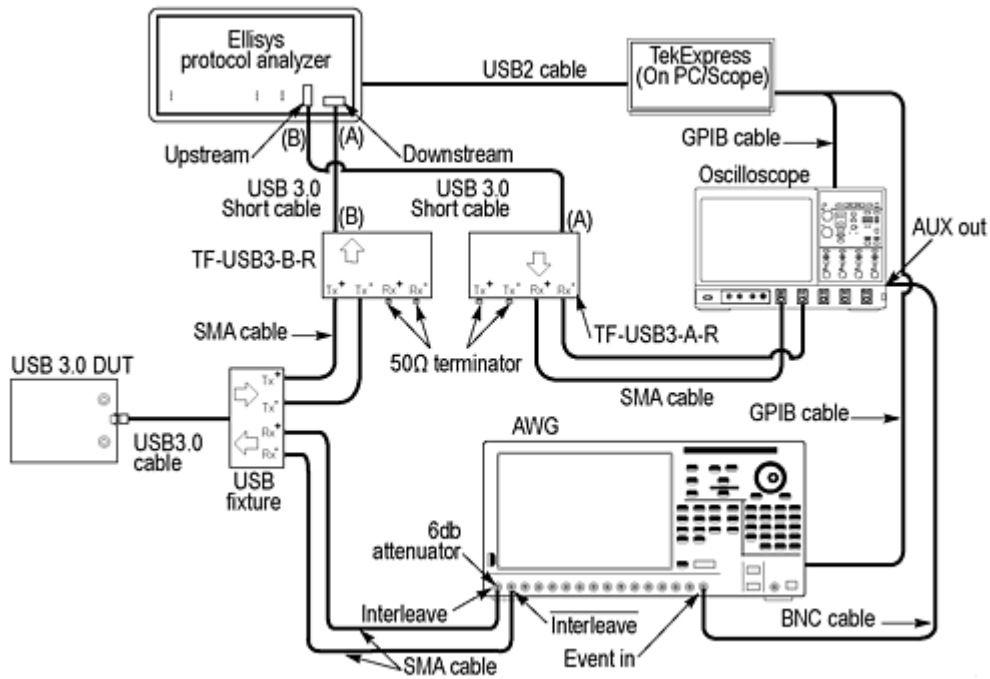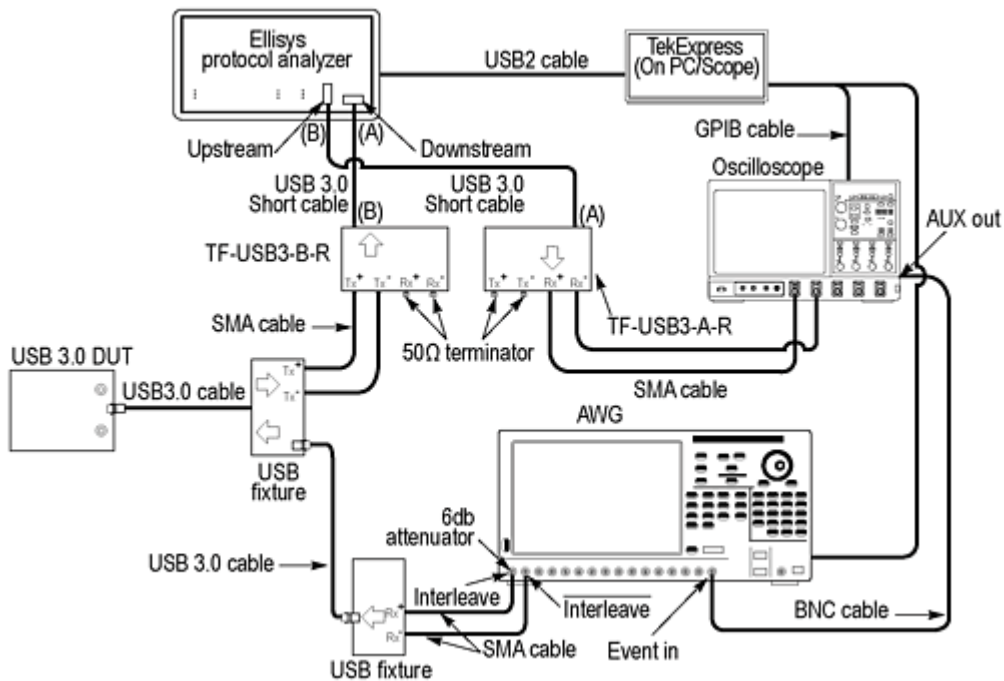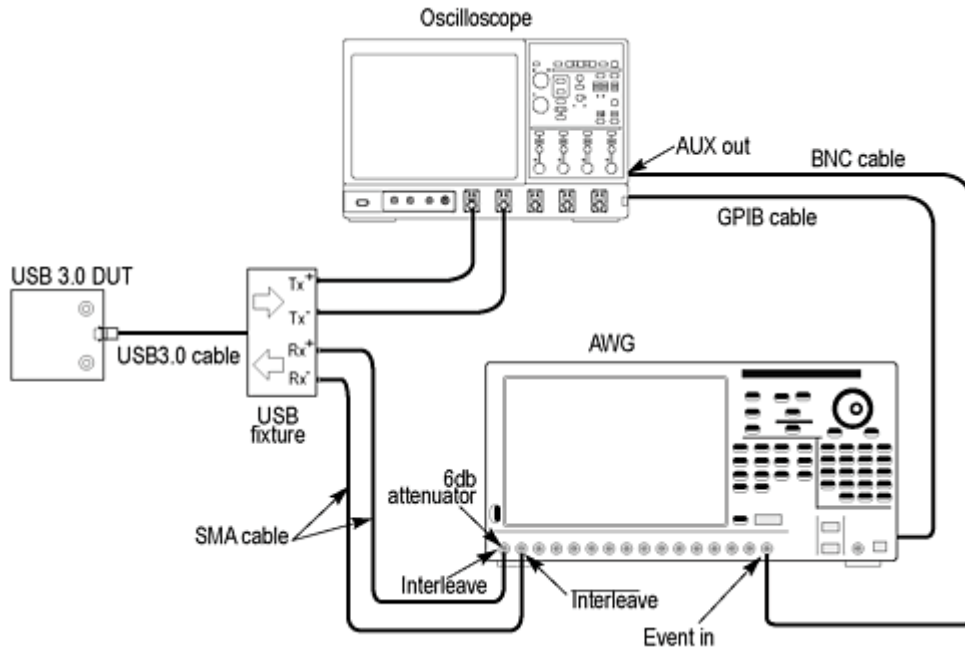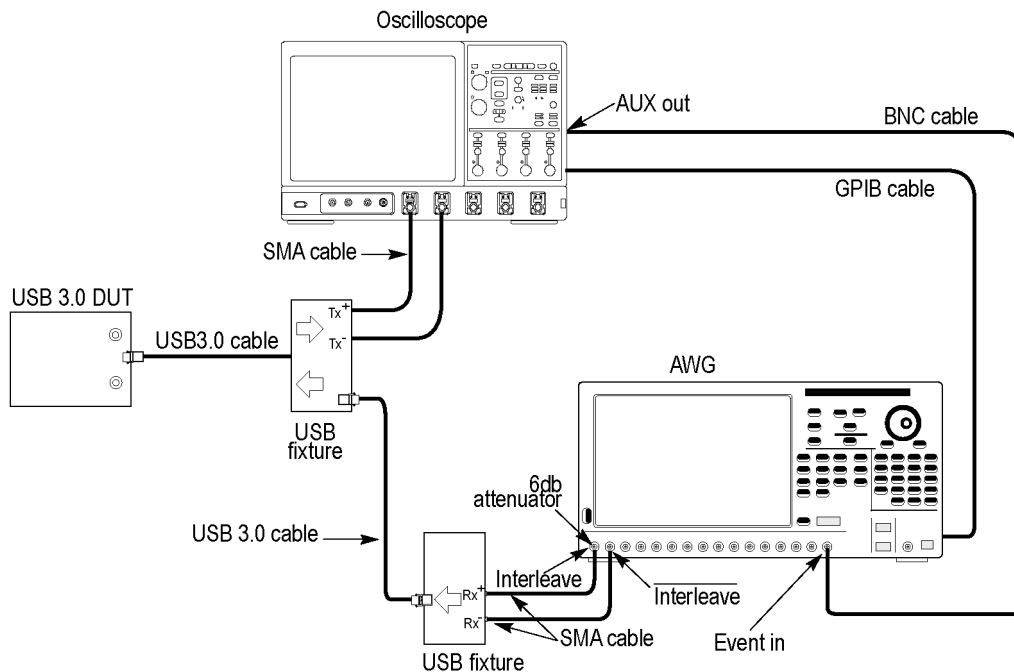
Schematics for Informative Test Approach

The following setup is required to test a device or a host using Informative Test approach:

> **NOTE.** *The schematics are common between Normative and Informative Tests. Refer to the "Schematics for Normative Test Approach" (see page 99) for more details.*

Device-Software Channel Emulation-Ellisys Error Detector (Using Oscilloscope to Measure Jitter)
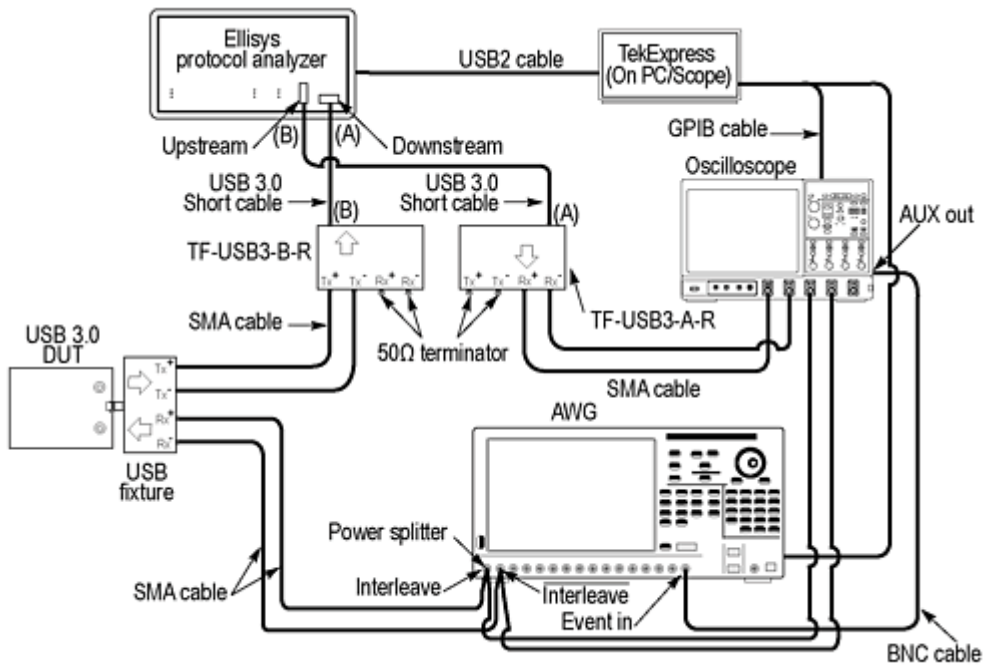
Device-Hardware Channel Emulation-Ellisys Error Detector (Using Oscilloscope to Measure Jitter)

Device-Software Channel Emulation-Oscilloscope Error Detector (Using Oscilloscope to Measure Jitter)

Device-Hardware Channel Emulation-Oscilloscope Error Detector (Using Oscilloscope to Measure Jitter)

Host-Software Channel Emulation-Ellisys Error Detector (Using Oscilloscope to Measure Jitter)
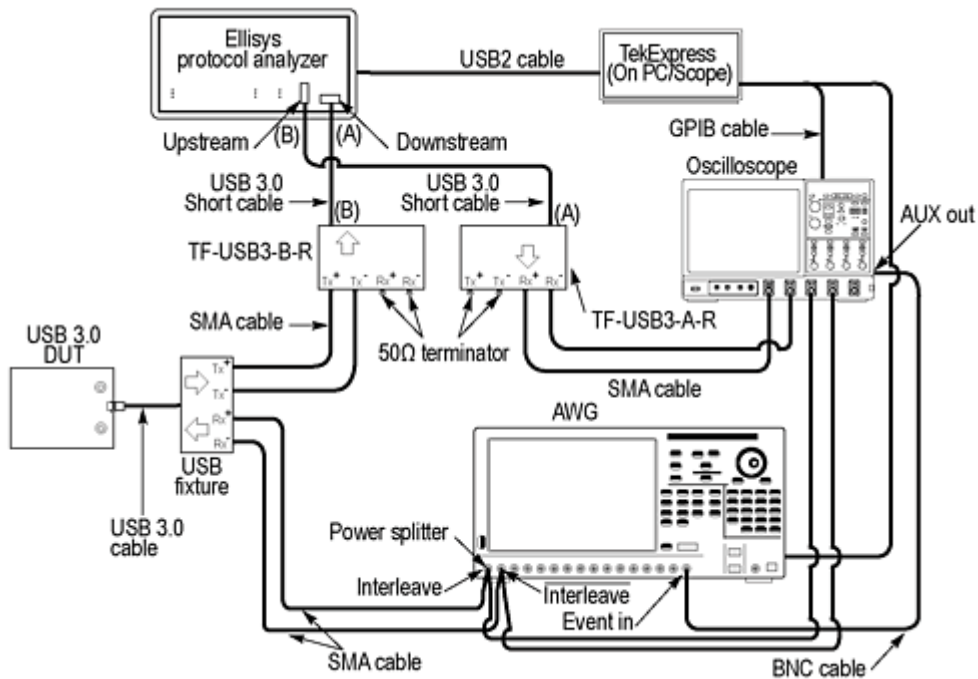
Host-Hardware Channel Emulation-Ellisys Error Detector (Using Oscilloscope to Measure Jitter)

Host-Software Channel Emulation-Oscilloscope Error Detector (Using Oscilloscope to Measure Jitter)

Host-Hardware Channel Emulation-Oscilloscope Error Detector (Using Oscilloscope to Measure Jitter)
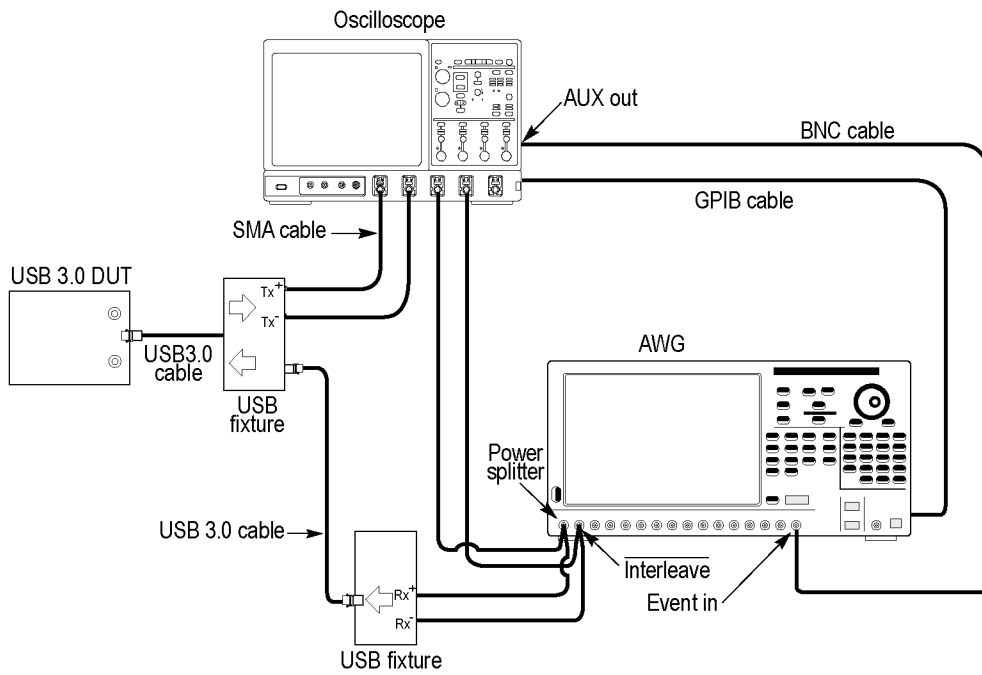
Ellisys Driver Installation

Follow the onscreen instructions as shown to install Ellisys driver:

Step 1:

Step 2:
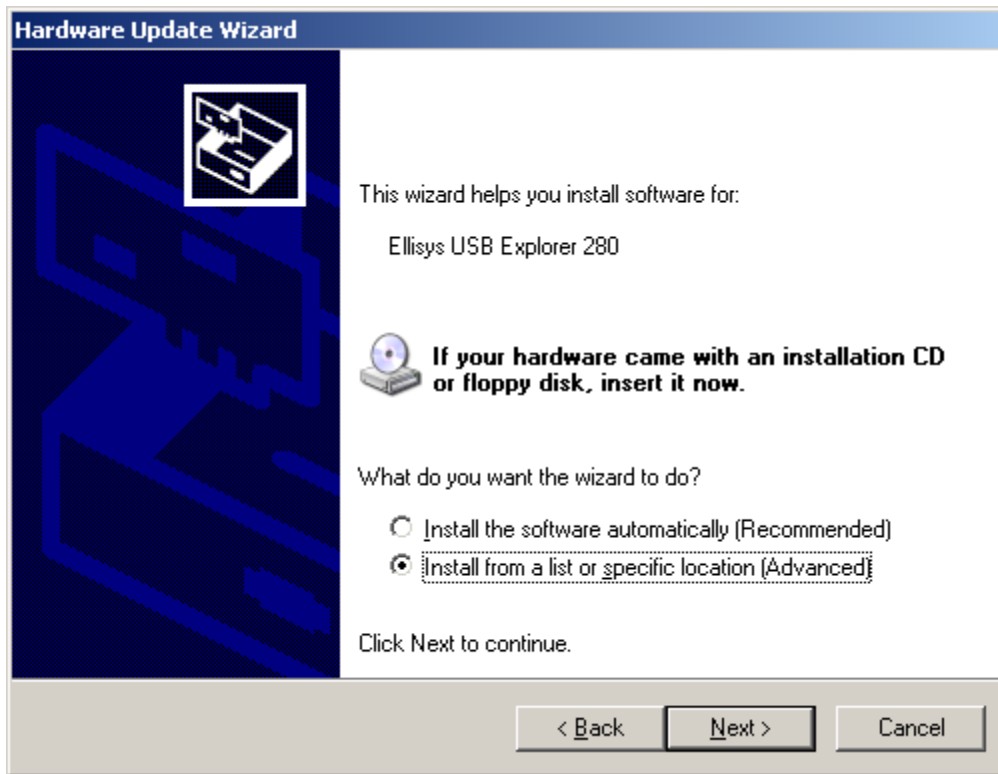
Step 3:
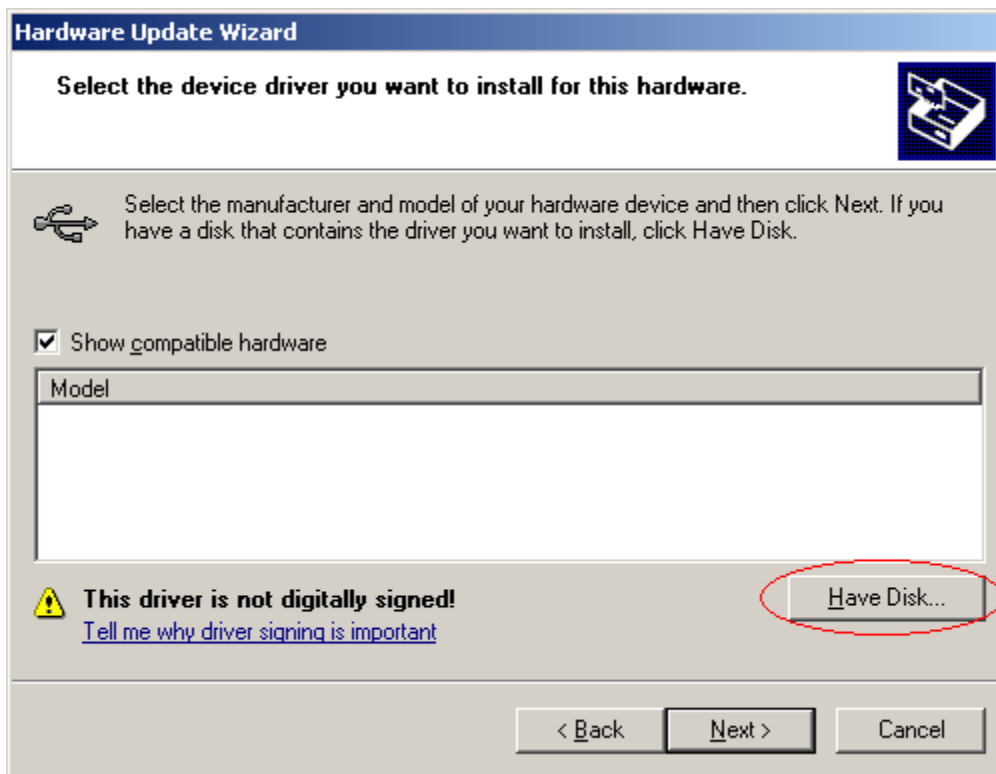
Step 4:
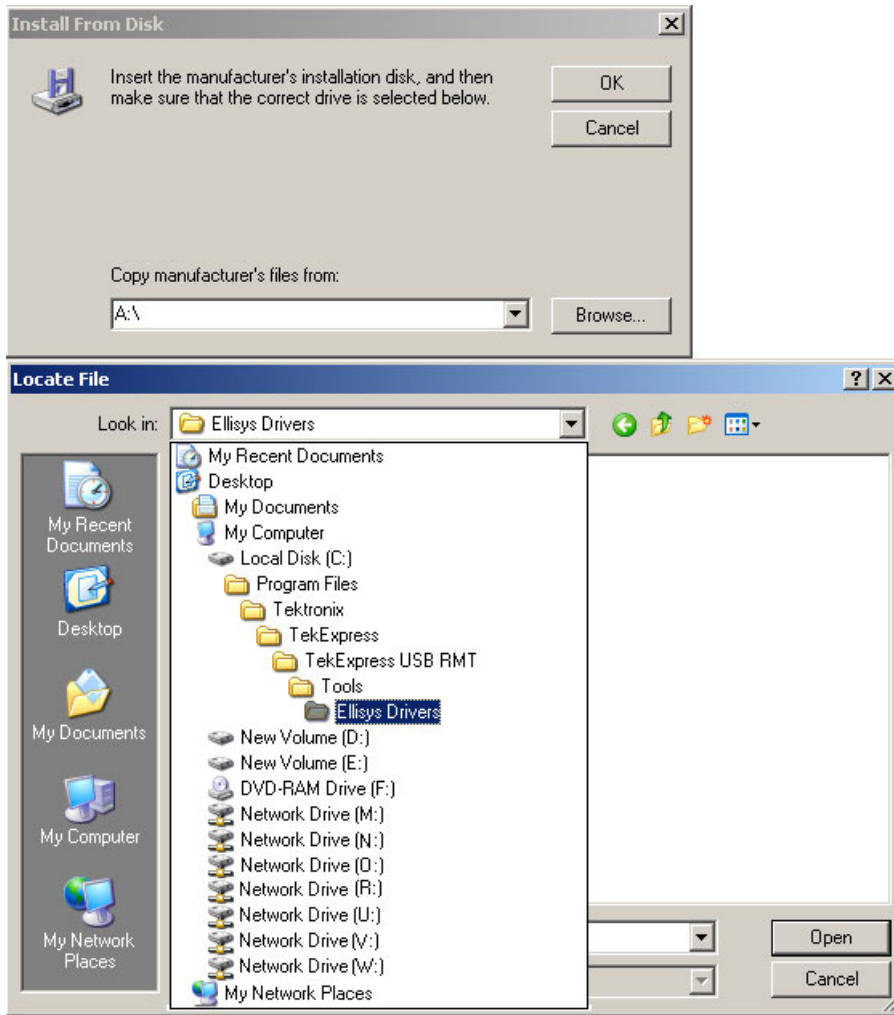
Step 5:
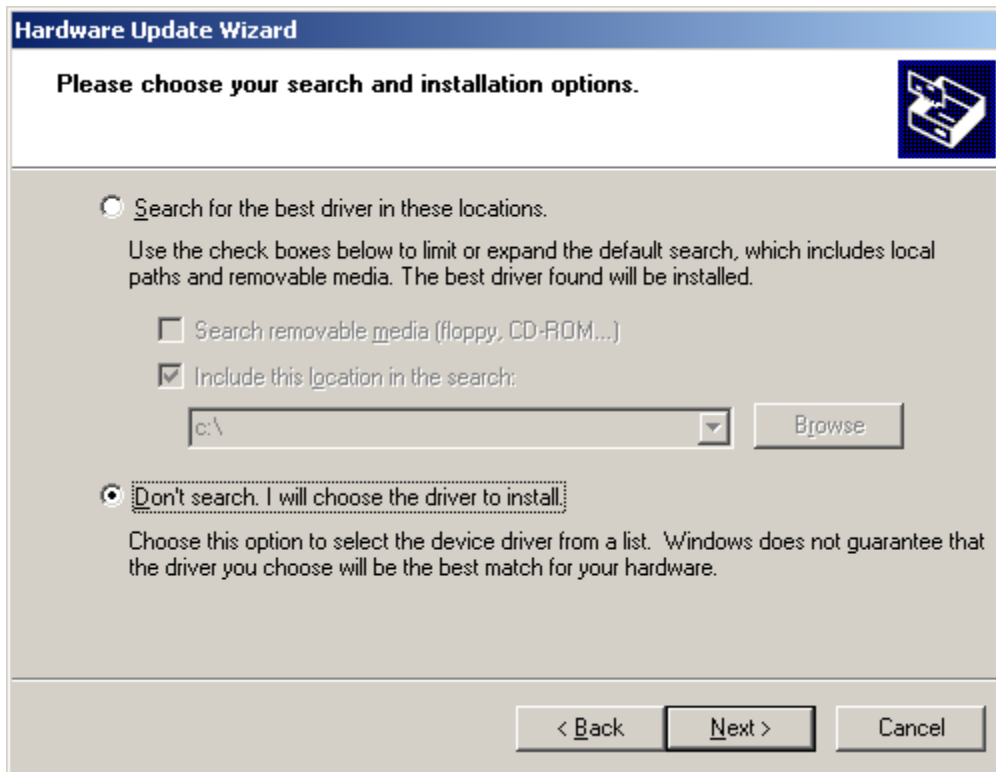
Step 6:
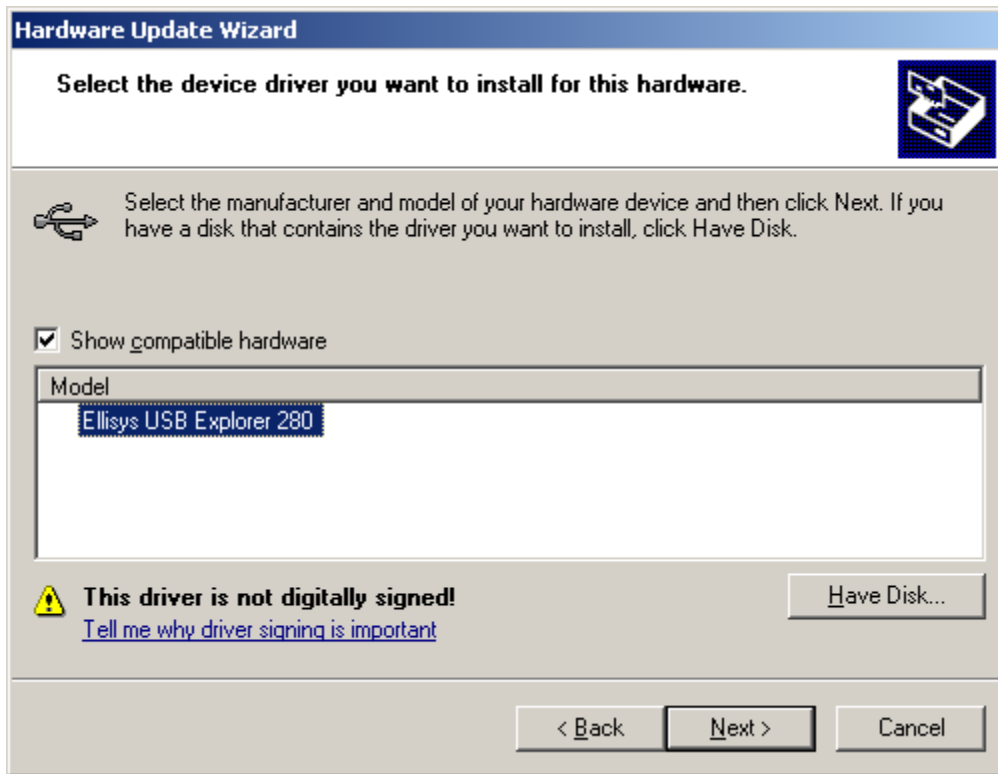
Step 7:
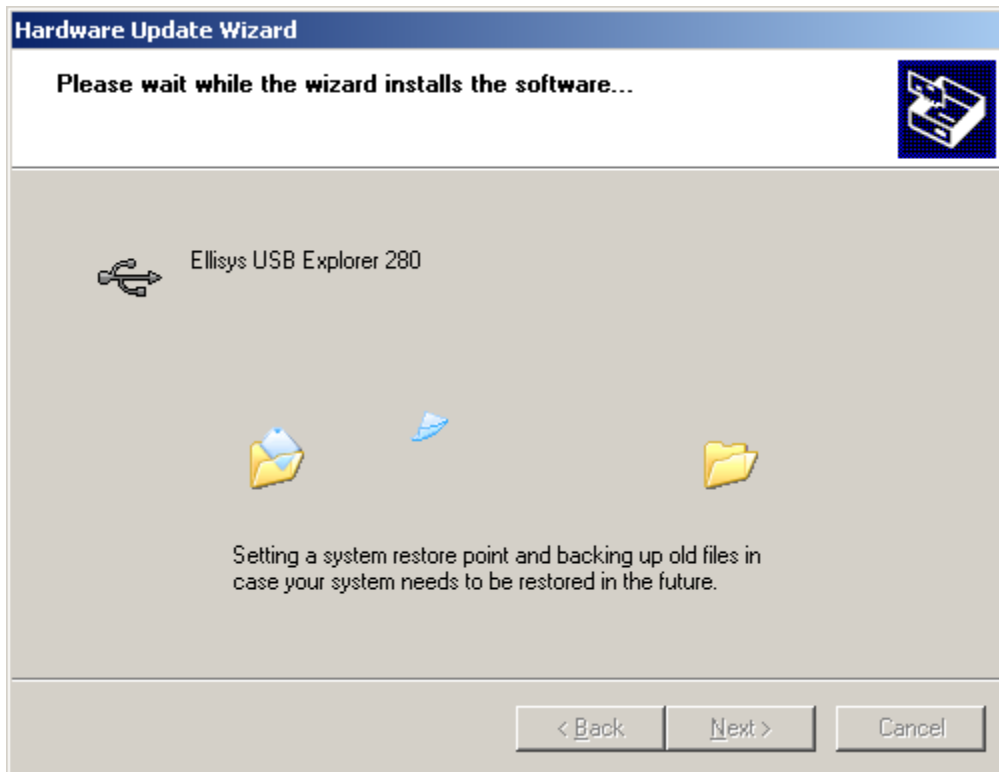
Step 8:
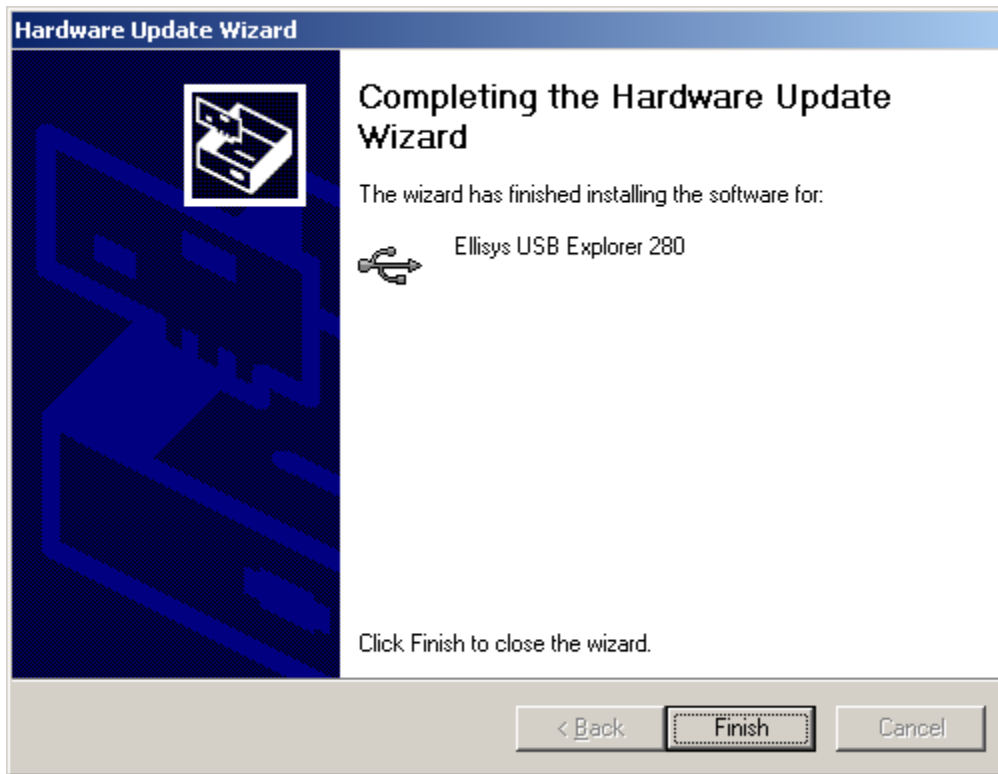
Shortcut Keys

The following table lists the short cut keys to access the application:

Table 32: Keyboard short cut keys

Menu	Shortcut keys
File	Alt + F
New Session	Ctrl + N
Open Session	Ctrl + O
Save Session	Ctrl + S
Save Session As	Alt + F + A
Save Report As	Alt + F + R
Print Preview Report	Alt + F + V
Print Report	Ctrl + P
Recent Sessions	Alt + F + E
Exit	Ctrl + X
View	Alt + V
Log File	Ctrl + L
Tools	Alt + T
Instrument Bench	Ctrl + I
Debug-Deskew Utility	Alt + T + D
Help	Alt + H
TekExpress Help (F1)	Alt + H + H
About	Alt + H + A

Error Codes for TekExpress

The following table lists the error codes for the application. Most of the errors require that you restart the system.

Table 33: Error codes and description

Error code	Description
<10000	TestStand generated error.
10001 - 11000 – Data Manager related errors	
10001	Insufficient Data. The Record could not be inserted. The following fields are empty or have insufficient data.
10002	Could not retrieve the record. The specified index is not valid.
11001 - 14000 – ICP related errors	

Table 33: Error codes and description (cont.)

Error code	Description
11001	Operation mode is not set as specified.
11002	SetUp file Error: Specified SetUp file is not set.
11003	Operation state is not set as specified.
11004	Specified waveform is not loaded into channel memory.
11005	Specified channel is not enabled.
11006	Interleave State could not be set to off.
11007	AWG Memory Import Error. Could not import the specified waveform into AWG memory. Filename not found.
11008	AWG Waveform Editing Error: Could not insert the specified pattern into the AWG sequence.
12001	Operation State is not set to required value.
12002	Display state is not set to required value.
12003	Horizontal Scale is not set to required value.
12004	Vertical scale is not set to required value.
12005	Vertical Position is not set to required value.
12006	Deskew is not set to specified value.
12007	Out of Range Error-RunTime Error Message.
13001	Acquisition timed out
13002	Channel deskew failed
13003	"Recall setup failed. Setup file not present in the specified location <File Path>
13004	Recall setup Failed - OperationTimed out
13005	Suspecting module failure\nFollowing channels does not have the required Peak-To-Peak value when TDR is generated\
13006	Please swap Module 1 with Module <N>
14001 - 18000 – SCP related errors	
14001	Timeout Error: Application could not be activated.
14002	JIT3 Application is already running.
14003	Cannot activate JIT3 application. Some other application is running on oscilloscope.
14004	Error recalling the specified setup.
14005	Error setting the specified Sequencer State.
14006	Error in closing the Application.
14007	Error loading the default setup.
15001 – 15006 – Iconnect Errors	
15001	Z-Line function failed\nOutput file not created
15002	S-Parameter function failed\nOutput file not created
15003	Eye measurement function failed\nOutput file not created
15004	Filter function failed\nOutput file not created

Table 33: Error codes and description (cont.)

Error code	Description
15005	IConnect not installed
15006	Input waveform not present in the specified location
18001	Error initializing Ellisys error detector.
20001 – 25000 – Framework related codes	
20001	Code for UI message passed from TestStand to Framework
20002	Code for UI message passed from TestStand to Framework
20003	Code for UI message passed from TestStand to Framework
20004	Code for UI message passed from TestStand to Framework
20005	Code for displaying dialog box passed from TestStand to Framework
20006	Code for displaying input box passed from TestStand to Framework
20007	Code for calling exception handler passed from TestStand to Framework
20008	Code for calling image box passed from TestStand to Framework
20009	Code for getting the pass/fail status in the analyze panel passed from TestStand to Framework
20010	Code for getting the prerecorded waveforms path passed from TestStand to Framework
20011	Code for displaying input dialog box passed from TestStand to Framework
20012	Code for getting the foot note passed from TestStand to Framework
25001 - 27000 - Other Errors	
25001	Analyze operation failed\n\nIConnect could not process the input waveforms or\nIt generates output waveforms that cannot be processed further\nCheck your configuration parameters
25002	LabVIEW Adapter configuration failed

Index