



**TELEDYNE LECROY**  
Everywhereyoulook™

# **PCI Express™ Script Automation Test Tool**

## **User Manual**

**for**

## **Teledyne LeCroy Analyzer/Trainer™**

**PCIe Protocol Suite software version 7.34**

Teledyne LeCroy Protocol Solutions Group  
Trademarks and Servicemarks

Teledyne LeCroy, CATC Trace, PETracer, PETracer Summit, Summit T3-16, and BusEngine are trademarks of Teledyne LeCroy.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Intel and Pentium are registered trademarks of Intel Corporation.

All other trademarks and registered trademarks are property of their respective owners.

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL INFORMATION, EXAMPLES AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE REPRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS ARE FULLY RESPONSIBLE FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN INFORMATION THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT TELEDYNE LECROY FOR A COPY.

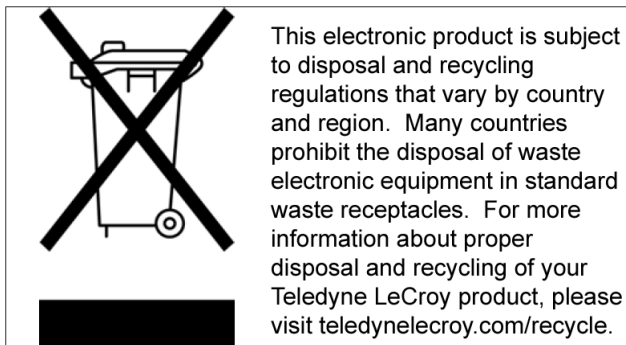
---

© 2012 Teledyne LeCroy, Inc. All rights reserved.

This document may be printed and reproduced without additional permission, but all copies should contain this copyright notice.

---

WEEE Program



Teledyne LeCroy  
3385 Scott Blvd.  
Santa Clara, CA 95054  
TEL: 800-909-7112 (USA and Canada)  
TEL: 408-653-1260 (worldwide)

# Contents

---

<b>Chapter 1: Introduction</b> .....	<b>1</b>
<b>1.1 PCI Express Automated Test Scripts</b> .....	<b>2</b>
<b>1.2 Summary of PCI Express Automated Tests</b> .....	<b>3</b>
1.2.1 Script Automation Test Tool .....	4
<b>1.3 Graphical User Interface</b> .....	<b>5</b>
<b>1.4 Test Information</b> .....	<b>8</b>
<b>1.5 Running Tests</b> .....	<b>12</b>
1.5.1 Endpoint DUT.....	12
1.5.2 Root Complex DUT .....	13
1.5.3 Switch Downstream Port DUT .....	13
1.5.4 Device Emulator Driver for Root Complex Testing .....	13
1.5.5 Installation Instructions under Linux.....	15
1.5.6 Module Removal .....	17
1.5.7 Test Driver Functionality (Windows or Linux) .....	17
<b>Chapter 2: PCI Express Tests</b> .....	<b>19</b>
<b>2.1 Endpoint Tests</b> .....	<b>19</b>
2.1.1 Link Layer Test Descriptions .....	19
2.1.2 Transaction Layer Test Descriptions .....	30
2.1.3 Tests Requiring DUT Actions .....	31
2.1.4 Root Complex Tests .....	34
2.1.5 Link Layer Test Descriptions .....	35
2.1.6 Transaction Layer Test Descriptions .....	44
<b>Chapter 3: Gen3 Compliance Testing</b> .....	<b>51</b>
<b>3.1 Hardware Requirements</b> .....	<b>51</b>
<b>3.2 Software Requirements</b> .....	<b>51</b>
<b>3.3 Software Installation</b> .....	<b>51</b>
<b>3.4 Hardware Installation</b> .....	<b>51</b>

---

<b>3.5 Using the Script Automation Test Tool .....</b>	<b>52</b>
<b>3.6 Tests .....</b>	<b>58</b>
<b>3.7 Verification of Test Compliance .....</b>	<b>59</b>
 <b>Appendix A: How to Contact Teledyne LeCroy.....</b>	 <b>61</b>

# Chapter 1

## Introduction

---

This document describes the operation of the Teledyne LeCroy PCI Express™ Script Automation Test Tool. The software includes tests for the Link Layer and Transaction Layer built to the guidelines defined by the PCI Special Interest Group (PCI SIG) specifications. The PCI Express Script Automation Test Tool operates on the Summit T3/Z3™, Summit T28/Z3™, and Summit T24/Z3™ platforms at x1, x2, x4, x8, and x16 lane widths. The *PETracer/Trainer* Summit™ Z2-16/T2-16 is also supported. The Teledyne LeCroy PCI Express Script Automation Test Tool supports Endpoint testing and Root Complex testing (and optionally supports Switch Downstream Port testing).

For both Endpoint Device and Root Complex testing, you must install PCIe Protocol Suite/*Trainer* software on the PC running the Script Automation Test Tool software and attach the *PETracer/Trainer* hardware to the USB ports on the PC. The PC must have one of the following operating systems: Windows 8 (x86 and x64), Windows Server 2012 (x64), Windows 7 (x86 and x64), Windows Server 2008R2 (x64) or Windows XP (x86), or one of the common Linux distributions (refer to the Linux driver installation instructions). The latest Service Pack available for the Windows OS in use is required. It is recommended that you use one of the supported 64-bit Windows versions listed above as they allow using more RAM than the 32-bit ones..

**Note:** You must do the following before using a network (Ethernet) attached Summit T3-16 Tracer and/or Summit Z3-16 Trainer with the Script Automation Test Tool. Run the PCIe Protocol Suite/*Trainer* software. Use the connection procedure to find and connect to the Tracer and/or Trainer device in the “All Connected Devices” dialog. In the “Connection Properties” dialog, displayed during the connection process, select **Automatically connect to the device**.

For Endpoint Device testing, place the device under test (DUT) into a *PETrainer*™ Host Emulation module. To emulate a root complex and exercise the DUT, connect the *PETrainer* Host Emulation module to a *PETrainer*. To capture and analyze traffic between the *PETrainer* and the DUT, also connect the *PETrainer* Host Emulation module to a *PETracer*.

For Root Complex (and Switch Downstream Port) testing, place a *PETrainer* Device Emulator card into the PCI Express slot in the Root Complex (or Switch Downstream Port) device under test. To emulate an Endpoint and exercise the DUT, connect the *PETrainer* Device Emulator card to a *PETrainer*. To capture and analyze traffic between the *PETrainer* and the DUT, also connect the *PETrainer* Device Emulator card to a *PETracer*.

For Root Complex testing, after you have completed the *PETracer/Trainer* setup described above, you must power up (boot) the DUT into the Microsoft Windows 8 (x86 and x64), Windows Server 2012 (x64), Windows 7 (x86 and x64), Windows Server 2008R2 (x64) or Windows XP (x86) OS, or one of the common Linux distributions. Turn on the PC. Open the **Script Automation Test Tool** software. Press the **Boot RC** button in the Script Automation Test Tool software, then turn on the DUT. You can then perform Root Complex testing. Install the Teledyne LeCroy *PETrainer* Device Emulator driver for the Device Emulator card (see instructions below).

The Teledyne LeCroy PCI Express Script Automation Test Tool software graphical user interface (GUI) allows you to select one or more PCI Express tests to run sequentially and to set property values such as lane width, polarity reversal; and log saving. The software automatically configures the *PETrainer* and *PETracer* systems. After test initiation, the log window shows test execution progress in real time. After completion of testing, the log window shows each Automated Test with a test result of either PASSED or FAILED.

## 1.1 PCI Express Automated Test Scripts

The PCI Express Automated Test scripts follow the test specification created by PCI SIG for manufacturers to use to assure a successful product. The test specification has five test areas:

**Link Layer Tests:** Device behavior for link level protocol.

**Transaction Layer Tests:** Device behavior for transaction level protocol.

**Configuration Space Tests:** Configuration space in PCI Express devices.

**Electrical Tests:** Platform and add-in card signal testing.

**Platform BIOS Tests:** Platform BIOS ability to recognize and configure PCI Express devices.

For each of these test areas, different tests are run for the:

Root Complex

Endpoint Device

Switch Downstream Port

Switch Upstream Port/Bridge

The PCI SIG test specifications identify tests that can be performed with automated test equipment. Teledyne LeCroy has developed some PCI Express Automated Test scripts for Link Layer and Transaction Layer Endpoint Device testing based on the following PCI SIG specifications:

Link Layer test scripts listed in section 4.0 of the PCI SIG specification:

**PCI\_Express\_Test\_Spec\_Link\_Layer\_2\_0\_rev\_1\_0\_17Sept2012\_Final.pdf**

Transaction Layer test scripts listed in section 5.0 of the PCI SIG specification:

**PCI\_Express\_Test\_Spec\_Transaction\_Layer\_2\_0\_rev\_1\_0\_11Aug2008.pdf**

Teledyne LeCroy has also developed PCI Express Automated Test Scripts for Root Complex testing.

The automated tests can be used for lane widths of x1, x2, x4, x8, and x16 on Summit T3/Z3™, Summit T28/Z3™, and Summit T24/Z3™. The *PETracer/Trainer Summit™ Z2-16/T2-16* is also supported.

## 1.2 Summary of PCI Express Automated Tests

The following Link Layer tests are supported:

### **Link Test 41-20 ReserveFieldsDLLPReceive Test**

Verify that DUT ignores reserved fields in an ACK DLLP.

### **Link Test 52-10 RetransmitOnNak Test**

Ensure that a DUT retransmits a transaction for which a NAK has been issued.

### **Link Test 52-11 REPLAY\_TIMER Test**

Ensure that a DUT REPLAY\_TIMER does not send an ACK or NAK.

### **Link Test 52-12 REPLAY\_NUM Test**

Ensure that a DUT keeps retransmitting a transaction for which a NAK has been issued on purpose until the number of times in REPLAY\_NUM.

### **Link Test 52-20 LinkRetrainOnRetryFail Test**

Ensure that the link connected to the DUT goes into retraining after trying and failing for REPLAY\_NUM of times to get a TLP through.

Check that the retry buffer and link states are not changed while in retraining.

### **Link Test 52-100 ReplayTLPOrder Test**

Verify that the oldest unacknowledged TLP is retransmitted first in replay, followed by the other unacknowledged TLPs in the same order in which they were initially transmitted.

### **Link Test 52-150 CorruptedCRC\_DLLP Test**

Ensure that a DUT recognizes a DLLP with a bad CRC, drops it, and logs a BAD\_DLLP port error.

### **Link Test 52-160 UndefinedDLLPEncoding Test**

Verify that the DUT silently drops any DLLP with undefined encoding and that no error is associated with it.

**Link Test 52-170 WrongSeqNuminAckDLLP Test**

Verify that the DUT drops any ACK DLLP that does not have a sequence number corresponding to an unacknowledged TLP and logs a BAD DLLP error associated with the port.

**Link Test 53-20 BadLCRC Test**

Verify that a receiver discards a TLP with a bad CRC by NAKing it and reports a BAD TLP error associated with the port.

**Link Test 53-31 DuplicateTLP Test**

Verify that the duplicate TLPs are handled properly by the DUT.

The following Transaction Layer tests are supported:

**Transaction Test 2-4 TXN\_BFT\_ErrorSignaling \***

Verify basic signaling functionality and message generation of a slotted Endpoint device.

**Transaction Test 3-1 TXN\_BFT\_FlowControlInit \***

Verify that a slotted Endpoint device receiver complies with basic flow control credit advertisement requirements.

**Transaction Test 5-1 TXN\_BFT\_VC0TCSupport \***

Verify that slotted Endpoint devices which do not support Virtual Channels beyond the default still handle requests with non-zero TC correctly.

**Transaction Test 1-1 TXN\_BFT\_RequestCompletion**

Verify basic Request and Completion handling of slotted Endpoint devices.

**Transaction Test 1-2 TXN\_BFT\_CompletionTimeout \***

Verify basic Completion Timer requirements of slotted Endpoint devices.

**Transaction Test 2-1 TXN\_BFT\_LegacyInt \***

Verify basic INTx message support requirements of slotted Endpoint devices.

\* These tests are optional.

**1.2.1 Script Automation Test Tool**

The PCI Express™ Script Automation Test Tool runs as a stand-alone (automated) application. The PCI Express Script Automation Test Tool uses the PCIe Protocol Suite Automation™ API to communicate commands to the *PETracer* and *PETrainer* platforms. It generates the stimulus for the test, records the traffic, and runs a Verification script to check the test result.

You can configure and operate a *PETracer™* Analyzer and *PETrainer™* Exerciser and sequentially run the tests described in this user manual. The PCI Express Script



Automation Test Tool starts the recording and then measures the performance characteristics of the DUT according to the standards defined by the appropriate test specification.

### 1.3 Graphical User Interface

The PCI Express™ Script Automation Test Tool application has a graphical user interface. This example shows the application running Endpoint tests.

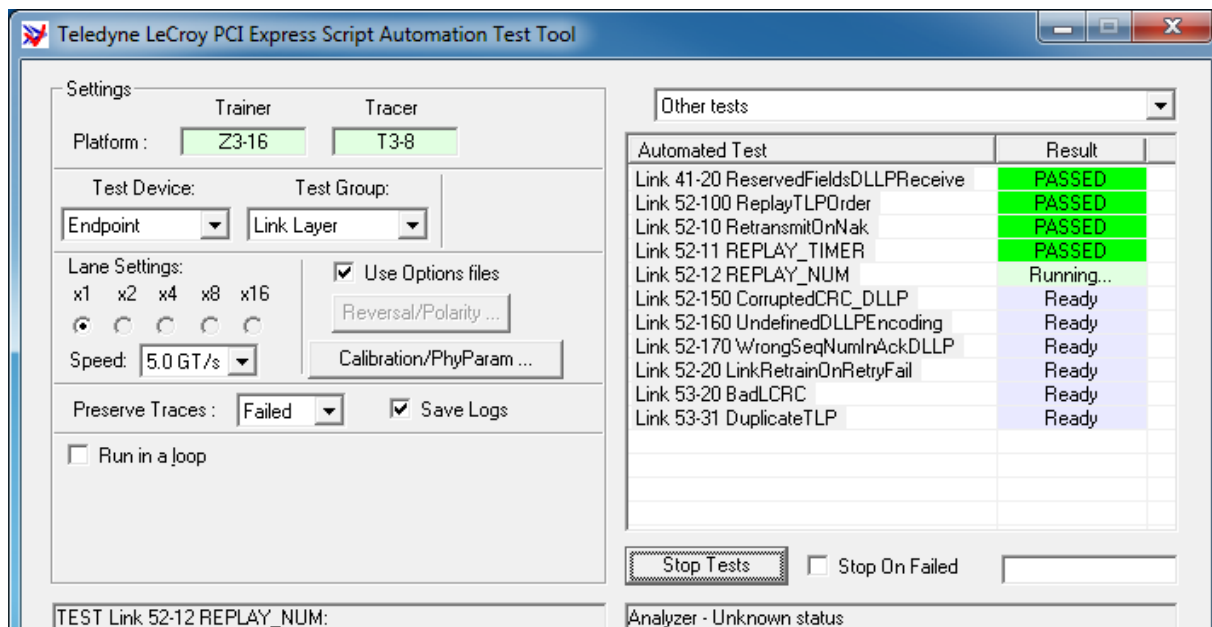


Figure 1.1: PCI Express Script Automation Test Tool Graphical User Interface.

The upper left Settings and Status areas show the following:

PETracer Analyzer and PETrainer Exerciser Platform: T3-16, T3-8, T28, T24, T2-16 for the Tracer platform, Summit Z2 or Summit Z3 for the Trainer platform [display only]

**NOTE:** Do not connect more than one trainer, only one is supported.

**Test Device:** From the list, select the device type to test. The two main device types supplied by Teledyne LeCroy are "Endpoint" and "Root Complex". You can create additional device types by specifying strings for the **TestDevice** variable in the Test Script File for user-defined tests (see the [Test Script Files<XREF>](#) section).

**Test Group:** From the list, select a group of tests for a Test Device. Test Group is a parameter specified in the Test Script File for each test (see the [Test Script Files<XREF>](#) section). The application sorts all Test Groups for a Test Device and adds them to the list. In addition to the defined groups, the list includes a group named "All", which has all tests for the Test Device.

**Lane Width:** Select x1, x2, x4, x8, or x16.

**Speed** (for the 5GT/s and 8GT/s licensed systems only): You can select **2.5 GT/s**, **5 GT/s**, **8GT/s** or **Auto** as the speed at which to run the tests. The **Auto** setting sets the speed for

recording to “Auto” and the speed of generation to whatever is defined in Generation Options and the Trainer script, allowing tests that change speed as they run. See the [Setting Test Speed<XREF>](#) section for how this setting affects the script and options files used in the tests.

**Use Options files:** Check to use the Lane Reversal and Polarity settings that you set in Recording Options and use the Generation Option files specified for tests. This includes the Automatic polarity settings. Uncheck to set Lane Reversal and Polarity settings manually.

**Reversal/Polarity:** Displays the dialog for manually specifying Lane Reversal and Polarity settings.

**Preserve Traces:** Select All, Failed, or None. If set to **Failed** or **All**, double-clicking a test in the list in the upper right of the window displays the CATC Trace™ of the test in the PCIe Protocol Suite application. Use the **Go to Marker** menu to locate failure points. Position the cursor over the vertical red bar indicating a marker to the left of the failed packet to read a tool tip with an explanation of why the PCI Express test failed. See below for an example of Link Layer Test 41-20 showing a marked failed packet and a tooltip explanation of the failure.

**Save logs:** Select to save the log files. A log file includes test results and traces for a test. The log files are stored in the **TestLog** folder of the **ScriptAutomationTestTool** folder. The name of the subfolder for each run of the tests is based on the date and time of the test run (for example, **06\_28\_2005\_\_15\_07**). An example log file folder is:

```
C:\Users\Public\Documents\LeCroy\PETracer\ScriptAutomationTestTool\TestLog\08_08_2014__10_41
```

See below for an example of a test log for a run of one Link Layer Test (41-20) for an Endpoint DUT in which the test passed. If a test fails, the test log identifies both the location of the failure (i.e., the packet number) and the reason for the test failure.

**Run in a loop:** You can run one or more tests repeatedly to accumulate loop results. Use loops to stress test all or some of the tests. After you select the **Run in a loop** option, the application allows you to specify the number of times to run the test cycle. If you leave the edit control empty or specify **0**, the loop runs until you press the **Stop Tests** button. As the loop cycles, the application will display the number of runs remaining and will count the passed and failed test cycles.

The upper right of the window lists the available tests and their status or results. The bottom half of the window displays in real time the status of the test, the log, and the test results.

Before running tests, select one or more tests to run sequentially:

Select a test and place it in the sequence of tests:

Position the cursor over a test, then press the **Shift** key and an “**up arrow**” or “**down arrow**” key.

Alternatively, press the **Shift** (for down) or **Ctrl** (for up) key and left-click the mouse.

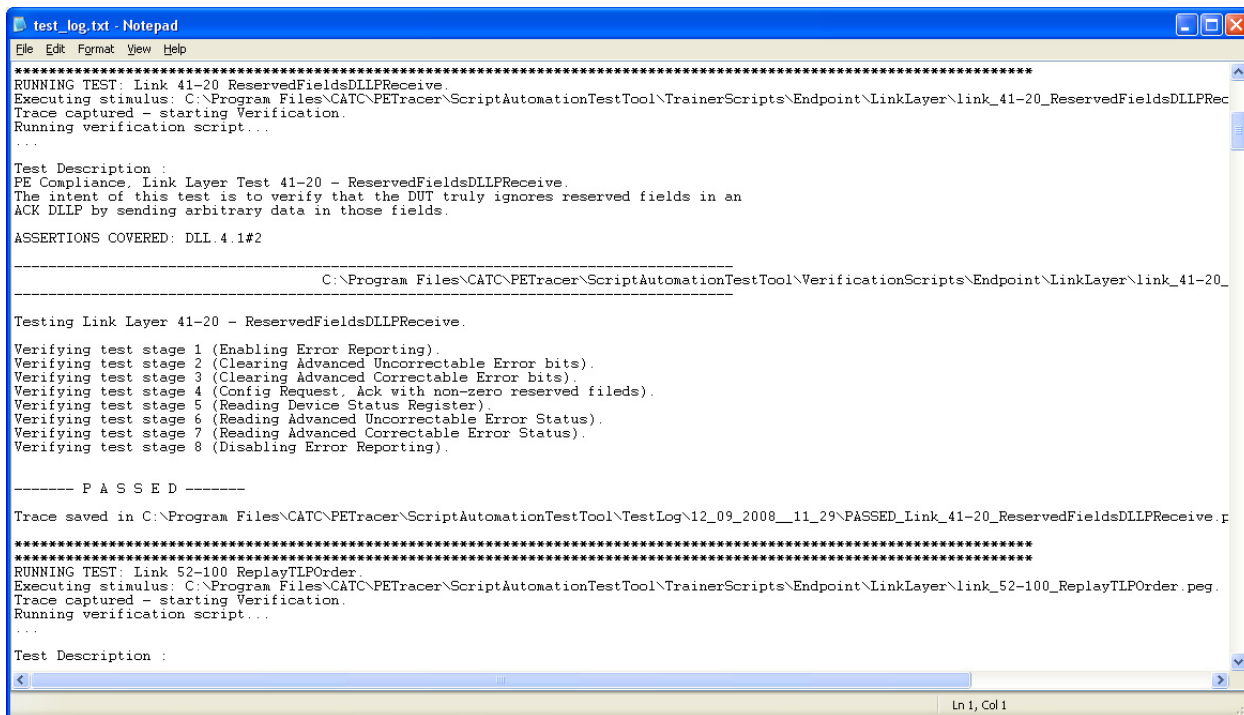
You can also left-click the test and drag the test up or down.

Selected tests have a shaded background. Unselected tests have no background. All selected tests should be in the top of the list with no unselected tests among them. Unselected tests should be last in the list.

To initiate testing, select the **Run Tests** button. The tests run automatically.

The upper right of the window shows test results (either PASSED in green or FAILED in red) for each selected test, (see the window example on the previous page).

**Note:** To terminate testing before all selected tests have been run, click the **Stop Tests** button.



```

test_log.txt - Notepad
File Edit Format View Help
*****
RUNNING TEST: Link 41-20 ReservedFieldsDLLPReceive.
Executing stimulus: C:\Program Files\CATC\PETracer\ScriptAutomationTestTool\TrainerScripts\Endpoint\LinkLayer\link_41-20_ReservedFieldsDLLPRec
Trace captured - starting Verification.
Running verification script...
...

Test Description :
PE Compliance, Link Layer Test 41-20 - ReservedFieldsDLLPReceive.
The intent of this test is to verify that the DUT truly ignores reserved fields in an
ACK DLLP by sending arbitrary data in those fields.

ASSERTIONS COVERED: DLL.4.1#2

-----
C:\Program Files\CATC\PETracer\ScriptAutomationTestTool\VerificationScripts\Endpoint\LinkLayer\link_41-20_
-----

Testing Link Layer 41-20 - ReservedFieldsDLLPReceive.

Verifying test stage 1 (Enabling Error Reporting).
Verifying test stage 2 (Clearing Advanced Uncorrectable Error bits).
Verifying test stage 3 (Clearing Advanced Correctable Error bits).
Verifying test stage 4 (Config Request, Ack with non-zero reserved fields).
Verifying test stage 5 (Reading Device Status Register).
Verifying test stage 6 (Reading Advanced Uncorrectable Error Status).
Verifying test stage 7 (Reading Advanced Correctable Error Status).
Verifying test stage 8 (Disabling Error Reporting).

----- P A S S E D -----

Trace saved in C:\Program Files\CATC\PETracer\ScriptAutomationTestTool\TestLog\12_09_2008__11_29\PASSED_Link_41-20_ReservedFieldsDLLPReceive.p
*****
RUNNING TEST: Link 52-100 ReplayTLPOrder
Executing stimulus: C:\Program Files\CATC\PETracer\ScriptAutomationTestTool\TrainerScripts\Endpoint\LinkLayer\link_52-100_ReplayTLPOrder.peg.
Trace captured - starting Verification.
Running verification script...
...

Test Description :

```

Figure 1.2: Test Log for a Link Layer Test (41-20) for an Endpoint DUT.

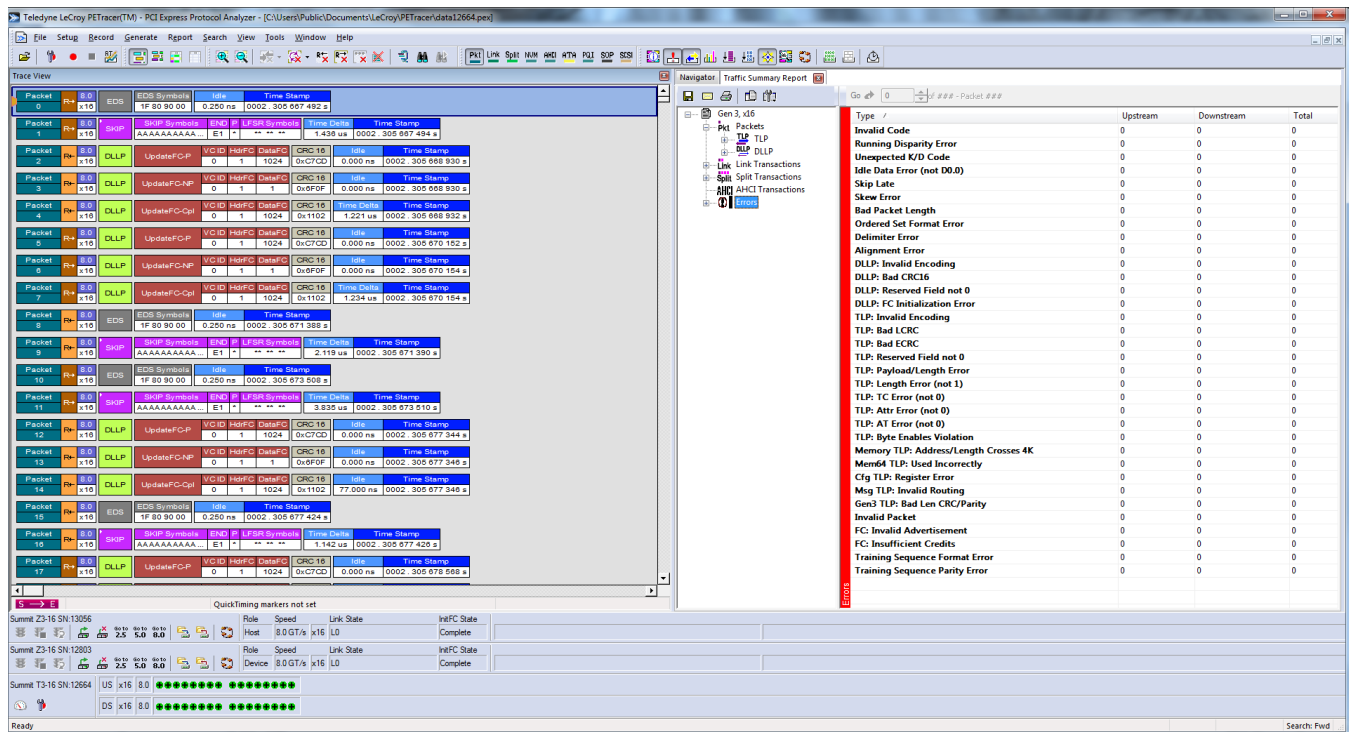


Figure 1.3: CATC Trace in PCIe Protocol Suite for a Link Layer Test (41-20) for an Endpoint DUT.

## 1.4 Test Information

The PCI Express Script Automation Test Tool uses the *PETracer/Trainer* Automation API to run the tests. The *PETracer* Automation™ API loads required Generation Options into *PETrainer*, loads required Recording Options into *PETracer*, generates test stimuli from the *PETrainer* script file, records the traffic, and runs a Verification script against the recorded trace to determine if the test PASSED or FAILED.

Each of the tests has the following information in its subfolders:

test script files

Recording Option settings file (.rec file)

Generation Option settings file (.gen file)

*PETrainer* Trainer Script (.peg file) for the test

*PETracer* Verification Script (.pevs file and its corresponding .inc files) for the test

### Test Script Files

All the components for a specific test are supplied in a test script file. Each test script file represents one test. The test script files for Endpoint Device testing are in the **Endpoint** subfolder of the **AutomatedTests** folder. The test script files for Root Complex testing are in the **RootComplex** subfolder of the **AutomatedTests** folder. The format of a test script file is:

```

TestAnalyzer = PETracer;
TestGroup   = "Link Layer";
TestName    = "Link 41-20 ReservedFieldsDLLPReceive";
TestCode    = "Test 41-20";

TestDescription = "The intent of this test is to verify that the DUT truly ignores reserved
fields in an ACK DLLP by sending arbitrary data in those fields.";

RecordingOptions = "RootComplex\\link_layer.rec";
GenerationOptions = "RootComplex\\link_layer.gen";
TrainerScript = "RootComplex\\LinkLayer\\link_41-20_ReservedFieldsDLLPReceive.peg";
VerificationScript = "RootComplex\\LinkLayer\\link_41-
20_ReservedFieldsDLLPReceive.pevs";

GenerationTimeout = 4000; // Optional parameter, tells the application to stop
// generation and recording after specified time in
// milliseconds in case generation didn't progress through
// all the stages and stopped in its own course.
// The default value of the timeout is 2 seconds.

TestDevice = "Root Complex"; // Optional parameter, specifies the DUT type.
// Default value is "Endpoint", specifying the Endpoint DUT

TestGenerator = "All"; // Optional parameter, specifies the type of LeCroy
// PCI Express Trainer product that must be used with this
// automated test. The possible values are:
// "All"
// "Z2-16"

```

You can add tests to the package to expand test coverage. To add a test, copy a similar existing test script file into any text editor. Replace the TestName and other information. Replace the Generation Options path, Recording Options path, Trainer Script path, and Verification Script path. Save as a new test script file in the **Endpoint** or **RootComplex** subfolder, depending on the DUT type, of the **AutomatedTests** folder. Ensure that the Generation Options file, Recording Options file, Trainer Script file, and Verification Script file are in the appropriate folders.

Please refer to PCIe Protocol Suite/*Trainer* documentation for information on how to create Recording and Generation Options and Trainer and Verification Scripts. You can use the Trainer and Verification Scripts included with the PCI Express Script Automation Test Tool to build your own tests.

Please refer to the 2.3 [Folder Structure<XREF>](#) section for specific file locations.

## Special TestGroup

Besides the "Link Layer" and "Transaction Layer" TestGroup, there is a TestGroup called **Special**. This TestGroup includes all tests that are executed before running the actual test. These tests do not appear in the test selection list but are always executed first each time you press the **Run Tests** button. The Special tests read the configuration space of the DUT and create a definitions file for the test that defines the required register addresses. The definitions file is required to customize the Trainer and Verification Scripts for the DUT. The **AutomatedTests\Endpoint\special\_test.cts** and **AutomatedTests\RootComplex\special\_test.cts** files have examples of the Special tests.

## RootBoot TestGroup

Besides the "Special," "Link Layer," and "Transaction Layer" TestGroup, there is a TestGroup called **RootBoot**. This TestGroup includes a test that is executed for a Root Complex DUT when you press the **Boot RC** button. Sections 2.4 and 2.5 have details on running Root Complex tests.

## Custom Initialization for Endpoint Device Testing

Some Endpoint devices require special initialization before a test can be performed. Custom initialization may require a Configuration Write, Memory Write, and/or IO Write transaction. For example, IO Write special initialization is required to enable error reporting.

To initialize an Endpoint device, modify the file **TrainerScripts\Templates\device\_specific\_init.peg** to include **Configuration Write**, **Memory Write**, and/or **IO Write** transactions.

The **TrainerScripts\Templates\device\_specific\_init.peg** file includes commented sample code to illustrate special initialization. Use that sample code to modify the file for special Endpoint Device testing. At the end of the file, set the **VAR\_CURRENT\_SEQUENCE\_NUMBER** parameter to an appropriate value.

The PCI Express Script Automation Test Tool runs the **TrainerScripts\Templates\device\_specific\_init.peg** script after each Link Up before starting the actual test.

## Setting Test Speed

The Script Automation Test Tool application [Speed<XREF>](#) setting allows you to select the speed at which to run the test. This setting applies only to the 5GT/s licensed systems. For these platforms, the **TrainerScripts** folder has a **Speed** subfolder, which contains three generator scripts with names related to the three speed settings:

**speed\_2\_5.peg**: This empty script does not contain any instructions.

**speed\_5\_0.peg**: This script contains a speed switch to 5 GT/s and a wait for 100 milliseconds.

**speed\_8\_0.peg**: This script contains a speed switch to 8 GT/s and a wait for 100 milliseconds.

**speed\_auto.peg**: This empty script does not contain any instructions.

When you select a speed setting in the application, the application copies the script corresponding to the selected speed to the **Speed** subfolder and gives it the name **switch\_speed.peg**. This file name is included at the end of the common linkup script, so all Link/Transaction tests can use it. You can include this file name in any user-defined test in order to use the Speed setting.

When you select a speed setting in the application, the application also modifies the following parameters in the Recording Options and Generation Options for each test, before executing the test:

If **Speed = 2.5 GT/s**, Recording Options Speed is set to 2.5 GT/s, and Generation Options TS Data Rate is set to 2.5 GT/s.

If **Speed = 5.0 GT/s**, Recording Options Speed is set to 5.0 GT/s, and Generation Options TS Data Rate is set to 5.0 and 2.5 GT/s.

If **Speed = 8.0 GT/s**, Recording Options Speed is set to 8.0 GT/s, and Generation Options TS Data Rate is set to 8.0, 5.0 and 2.5 GT/s.

If **Speed = Auto**, Recording Options Speed is set to Auto, and Generation Options TS Data Rate is set to 8.0 and 5.0 and 2.5 GT/s.

## Folder Structure

The PCI Express Script Automation Test Tool uses a specific folder structure. The top folder is:

```
C:\Users\Public\Documents\LeCroy\PETracer\ScriptAutomationTestTool\
```

...which has the following required subfolders:

- AutomatedTests
- RecordingOptions
- GenerationOptions
- TrainerScripts
- Verification Scripts

## AutomatedTests Folder

Contains the test script files in the **Endpoint** (for Endpoint Device testing) or **RootComplex** (for Root Complex testing) subfolders.

## RecordingOptions Folder

Contains recording option files in the **T2** (for *PETracer* T2 **.rec** files) or **T3** (for *PETracer* T3 **.rec** files) subfolders. The T2 and T3 subfolders each have **Endpoint** (for Endpoint Device testing) and **RootComplex** (for Root Complex testing) subfolders. For Endpoint Device testing, place a *PETracer* T2 recording options file in **T2\Endpoint** and a *PETracer* T3 recording options file of the same name in **T3 \Endpoint**. For Root Complex testing, place a *PETracer* T2 recording options file in **T2 \RootComplex**, and a *PETracer* T3 recording options file of the same name in **T3 \RootComplex**. Enter the file path in the test script file RecordingOptions line, for example,

```
RecordingOptions = "Endpoint\\user_test.rec";
```

## GenerationOptions Folder

Contains generation option files. This folder has the same subfolders as the RecordingOptions folder. Enter the file path in the test script file GenerationOptions line, for example,

```
GenerationOptions = "Endpoint\\user_test.gen";
```

## TrainerScripts Folder

Contains *PETrainer* script (**.peg**) files. You can use any subfolder structure. Enter the file path in the test script file TrainerScript line, for example,

```
TrainerScript = "RootComplex\\LinkLayer\\link_41-20_ReservedFieldsDLLPReceive.peg";
```

The path is relative from the TrainerScripts folder. In this example, a subfolder named **LinkLayer** is under the **RootComplex** folder in the **TrainerScripts** folder and contains the **link\_41-20\_ReservedFieldsDLLPReceive.peg** script.

## Verification Scripts Folder

Contains *PETracer* Verification Script (**.pevs**) files (and their corresponding **.inc** files). You can use any subfolder structure. If **.inc** files are included in a Verification Script, reference them by the relative path from the Verification Script location. Enter the file path in the test script file VerificationScript line, for example:

```
VerificationScript = "Endpoint\\LinkLayer\\link_41-20_ReservedFieldsDLLPReceive.pevs";
```

In this example, a subfolder named **LinkLayer** is under the **Endpoint** folder in the **VerificationScripts** folder and contains the script **link\_41-20\_ReservedFieldsDLLPReceive.pevs**.

## 1.5 Running Tests

For the Endpoint or Root Complex DUT type, perform the following steps to execute a test.

### 1.5.1 Endpoint DUT

For Endpoint Device testing, perform the following steps to execute a test:

Turn OFF the *PETrainer* Host Emulation module.

Place the Endpoint DUT into the *PETrainer* Host Emulation module.

Connect the *PETrainer* Exerciser system to the *PETrainer* Host Emulation module using cables supplied by Teledyne LeCroy.

Connect the *PETracer* Analyzer system to the *PETrainer* Host Emulation module using cables supplied by Teledyne LeCroy.

Connect the *PETracer* and *PETrainer* systems to the USB ports of the PC on which the PCI Express Script Automation Test Tool is running.

Power on the PC and the *PETracer* and *PETrainer* systems.



Open the PCI Express Script Automation Test Tool and select **Endpoint** for the DUT Type in the GUI.

Turn ON the *PETrainer* Host Emulation module.

**Note:** If the DUT card requires external power to run, attach the power supply to it and turn it on.

Select Endpoint tests to execute.

Click the **Run Tests** button to begin testing.

**Note:** If the Special test to scan and process the DUT configuration space fails, press the **Reset** button on the *PETrainer* Host Emulation module to try again.

### 1.5.2 Root Complex DUT

For Root Complex testing, perform the following steps to execute a test:

1. Turn OFF the DUT PC.
2. Insert the *PETrainer* Device Emulator card in the slot to be tested on the DUT system.
3. Connect the *PETrainer* Exerciser system to the *PETrainer* Device Emulator card using cables supplied by Teledyne LeCroy.
4. Connect the *PETracer* Analyzer system to the Device Emulator card using cables supplied by Teledyne LeCroy.
5. Connect the *PETracer* and *PETrainer* systems to the USB ports of the PC on which the PCI Express Script Automation Test Tool is running.
6. Power ON the PC and the *PETracer* and *PETrainer* systems.
7. Open the PCI Express Script Automation Test Tool and select **Root Complex** as the DUT Type in the GUI.
8. Click the **Boot RC** button in the PCI Express Script Automation Test Tool.
9. Turn ON the DUT PC.
10. Boot the DUT PC into the Windows 8 (x86 and x64), Windows Server 2012 (x64), Windows 7 (x86 and x64), Windows Server 2008R2 (x64) or Windows XP (x86) OS, or to one of the common Linux distributions.
11. Install the Teledyne LeCroy *PETrainer* Device Emulator driver for the Device Emulator card using the installation instructions below.
12. Select Root Complex tests to execute.

### 1.5.3 Switch Downstream Port DUT

The Root Complex setup can also be used to test a downstream port of a Switch device.

Insert the Switch interface card in the PCI Express slot of the host PC. Insert the Device Emulator card in the downstream slot on the Switch.

Otherwise, perform the same steps as in 2.4.2 Root Complex DUT.

### 1.5.4 Device Emulator Driver for Root Complex Testing

Root Complex testing requires installing a driver for the Device Emulator PCI Express Endpoint on the DUT system. Currently supported operating systems are:

Windows 8 (x86 and x64), Windows Server 2012 (x64), Windows 7 (x86 and x64) (see details below), Windows Server 2008R2 (x64) or Windows XP (x86) OS (**lcrypedr.inf** driver)

Ubuntu 14.10, CentOS 7 (see details below)

After installing the PETracer application, you can find the drivers at \Public Documents\Lecroy\Petracer\ScriptAutomationTestTool\Drivers folder. Copy the **\ScriptAutomationTestTool\Drivers** folder from the testing PC to a CD to use for installation on the DUT.

The **\Drivers** folder contains three subfolders:

**\Xp** contains the **lcrypedr.inf** driver for the Windows XP operating system.

**\Server2003** contains the **lcrypedr.inf** driver for the Windows Server 2003 operating system.

**\Win7** contains the **lcrypedr.inf** driver for the Windows 7 operating system.

**\Linux** contains the files needed to install and build the driver under Linux distributions.

### Installation Instructions under Windows

When you first boot the system for Root Complex testing, the **New Device** plug-and-play wizard appears for the Device Emulator PCI Express Endpoint. The Device Emulator presents itself to the system as an Ethernet Controller device. Follow the wizard instructions. When asked for the driver location, point to the appropriate subfolder of the **\Drivers** folder for your operating system.

The **lcrypedr.inf** driver installs. The Device Emulator device appears in the Device Manager under a **Sample Drivers** group named **LeCroy PETrainer Device Emulator Driver for PCI Express Root testing**.

After you install the driver, you can restart the DUT, if necessary, and perform testing.

If you move the Device Emulator to another PCI Express slot for testing, you must install the driver again. After the first test, you can use the **Install the software automatically** option of the **New Device** plug-and-play wizard to install the driver.

### Note on enabling the “phantom” Advanced Error Reporting Capability

The Advanced Error Reporting Capability structure for the Root or Switch ports should enable by default. However, some Root Complex DUT setups may not enable the Advanced Error Reporting Capability structure for the Root or Switch ports. Advanced Error Reporting may be fully operational, but the Advanced Error Reporting Capability structure does not link within the Root or Switch port’s Enhanced Configuration Space.

The Device Emulator Driver includes an option to test Advanced Error Reporting features. After you install the driver, perform the following steps to perform the test:

1. Using a registry editor, locate the enumerator key for the Device Emulator under the **HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Enum\PCI** path. The name is based on the LeCroy Vendor ID (0x1570) and the Product ID used for Root testing (0xA239).
2. Under the enumerator key, find the key for the current device instance. That key includes a **Control** entry.
3. Open the **DeviceParameters** folder for the instance key.
4. Add a DWORD value named **AdvancedErrorStructOffset** in the **DeviceParameters** folder. Set the value for this entry to the byte offset of the “phantom” Advanced Error Reporting Capability structure within the Root or Switch port’s Enhanced Configuration Space.
5. Re-start the DUT.

You can now perform full testing of the port, including Advanced Error Reporting.

## 1.5.5 Installation Instructions under Linux

### Dependencies

#### Root Access

Installation of the module requires root privileges. Preferably, grant **sudo** access to normal users:

```
$ su -c "vi /etc/sudoers" (or editor of choice)
```

(You can also log in as the root user, or switch to **root** via the **su** command.)

Add individual user access by adding your username and permissions. For example:

```
<username> ALL=(ALL) ALL
```

You can also add group access by uncommenting and/or modifying one of the provided examples.

The **sudoers** file describes the various permissions.

### GNU C Compiler (GCC)

Only a GNU C compiler can compile the Device Emulator module source code. Some Linux distributions install this compiler by default. To check whether your Linux distribution has installed GCC, type **gcc** to see if the command is found or not.

Ubuntu typically installs GCC by default.

### Linux Kernel Headers

To compile the module for the particular kernel requires kernel headers. The kernel header version should match that of the kernel. The package management system usually manages kernel headers.

## Ubuntu packages

When you are building the Root Complex test driver using Ubuntu distribution (latest official version used: 14.10) of Linux ensure that you have refreshed your libraries and added necessary modules/packages to the installation:

Run:

- a. `sudo apt-get install gcc`
- b. `sudo apt-get update`
- c. `sudo apt-get dist-upgrade (ONLY If applies)`

## Module Use

### Source Installation

Choose a directory in which to place the module sources, change to that directory, and "untar" the source tarball.

```
$ tar xzf pcide_1_0.tar.gz
$ cd pcide
$ make
```

### Module Installation

You must have root privileges to install the module. Preferably, use the **sudo** command to install the module with root privileges. During installation, you can pass optional commands to control the module.

```
$ sudo insmod pcide.ko <option=0|1>
```

The available options are:

poll=1	<p><b>poll=1</b> is for module to poll Device Emulator commands.</p> <p><b>poll=0</b> is for module to use an interrupt.</p> <p>If you omit this parameter, default is 0.</p> <p><b>Important:</b> Use the poll=1 polling mode when testing with the Summit Z3 and Z2 Trainer traffic generators.</p>
msi=1	<p><b>msi=1</b> is for MSI mode interrupts.</p> <p><b>msi=0</b> is for legacy interrupts. Legacy interrupts are ignored if you use the <b>poll=1</b> option.</p> <p>If you omit this parameter, default is 0.</p> <p>Note: This setting is currently not in use.</p>
debug=1	<p><b>debug=1</b> prints verbose debug messages to the system or kernel logging file.</p> <p><b>debug=0</b> prints normal messages.</p> <p>If you omit this parameter, default is 0.</p>
mcfg=<ADDRESS >	<p>Use decimal or hexadecimal (prefixed with "0x") for the root port configuration space address. The system uses this address only if the module does not find the value.</p>

aer=<ADDRESS>	Use decimal or hexadecimal (prefixed with "0x") for the root port advanced error reporting space address. The system uses this address only if the module does not find the value.
---------------	--

An example of module installation with options is:

```
$ sudo insmod pcide.ko debug=1 poll=1
```

## 1.5.6 Module Removal

To remove the module, type:

```
$ sudo rmmmod pcide
```

Using the above methods, the module does not stay loaded in memory. You must install the module after each reboot.

To install the module so that it stays loaded in memory, you can:

- ❑ Call the **sudo insmod** ... command from a script run at boot time with root privileges.
- ❑ Copy the **pcide.ko** file into the path **/lib/modules/\$(uname -r)** where it will load automatically.  
**\$(uname -r)** will resolve into the particular kernel version number that is currently running.

## 1.5.7 Test Driver Functionality (Windows or Linux)

The driver does the following:

- ❑ Allocates a buffer in the system memory that is accessible by the Device Emulator for storing commands to the driver as well as for general DMA.
- ❑ Maps the Device Emulator memory and IO spaces for access.
- ❑ Locates the configuration space for the Root or Switch downstream port to which the Device Emulator attaches.
- ❑ Takes control of the named Root or Switch port and configures it so that no errors on that port result in an interrupt being forwarded to the system software.
- ❑ Implements a protocol with the Device Emulator that the CTP will use for executing the tests.

The driver locates the configuration space for the Root or Switch downstream port to which the Device Emulator attaches by following the ACPI structures created by the system BIOS. It looks for the **MCFG** structure (as defined by the PCI Firmware specification) to be pointed at by the ACPI **RSDT** or **XSDT** structures. The **MCFG** structure points to the location of the mapped PCI Express configuration spaces.

If the driver cannot find the **MCFG** structure, **RSDT** structure, **XSDT** structure, Device Emulator configuration space, or Root or Switch downstream port configuration space, the Script Automation Test Tool refuses to perform any of the normal tests. To indicate this case, the software reports that the Special test FAILED.

The Device Emulator communicates with the driver by:

1. Writing a command in the command area of the system buffer.
2. Interrupting the system using legacy PCI interrupt emulation.

Upon receiving the interrupt, the driver acknowledges it, examines the command buffer contents, and executes the command.

The command types that the driver can execute are:

- ❑ Report information about system variables (such as the physical address of the command buffer) and the root/switch port found. The Special test for Root testing uses this command (the default command).
- ❑ Execute a Configuration Read transaction from the PCI compatible portion of the Device Emulator configuration space.
- ❑ Execute a Configuration Write transaction to the PCI compatible portion of the Device Emulator configuration space.
- ❑ Execute a Memory Read transaction to one of the two Memory spaces on the Device Emulator.
- ❑ Execute a Memory Write transaction to one of the two Memory spaces on the Device Emulator.
- ❑ Execute an IO Read transaction to the IO space on the Device Emulator.
- ❑ Execute an IO Write transaction to the IO space on the Device Emulator.
- ❑ Clear all error status bits for the Root or Switch port to which the Device Emulator attaches.
- ❑ Reflect the current values of the Error Reporting Registers for the Root or Switch port to which the Device Emulator attaches to the PCI Express link. This can use Configuration Writes or Memory Writes.
- ❑ Retrain or Hard Reset the link to the Device Emulator.
- ❑ Send a series of Memory Write transactions. The Device Emulator monitors those transactions and signals when the driver should stop sending them. (This command can ensure that the next transaction from the DUT has a specific Tag value, so that it can be completed from the Trainer script.)
- ❑ Read a specified register from the Configuration Space for the Root or Switch port to which the Device Emulator attaches.
- ❑ Write a specified register to the Configuration Space for the Root or Switch port to which the Device Emulator attaches.

Chapter 3 refers to these commands when describing how Root Complex tests are performed.

# Chapter 2

## PCI Express Tests

---

### 2.1 Endpoint Tests

#### 2.1.1 Link Layer Test Descriptions

##### Test 41-20. ReservedFieldsDLLPReceive

**ASSERTIONS COVERED:** DLL.4.1#2

The intent of this test is to verify that the DUT truly ignores reserved fields in an ACK DLLP by sending arbitrary data in those fields.

**Trainer Stimulus:** link\_41-20\_ReservedFieldsDLLPReceive.peg

**Recording Options:** link\_layer\_common.rec

**Verification Scripts:** link\_41-20\_ReservedFieldsDLLPReceive.pevs

##### Test Algorithm:

1. Write 0x0000000F to Device Control Register, to enable Error Reporting for all error types.
2. Write 0xFFFFFFFF to Advanced Uncorrectable Error Status Register, to clear all prior error bits.
3. Write 0xFFFFFFFF to Advanced Correctable Error Status Register, to clear all prior error bits.
4. Main test stage. Do the following:
5. Send configuration read request TLP.
6. Wait for completion.
7. Send an Ack DLLP with non-zero reserved field.
8. Read Device Status Register. No error bits should be set.
9. Read Advanced Uncorrectable Error Status Register. No error bits should be set.
10. Read Advanced Correctable Error Status Register. No error bits should be set.
11. Write 0 to Device Control Register, to disable Error Reporting for all error types.

##### Pass/Fail Criteria:

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

- Verify that:
- The DUT ignored the non-zero field(s) in the ACK and did not retransmit the CONFIG\_RD\_COMPLETION TLP.
- The DUT did not set any error bit in its Device Status register and did not generate any error messages.
- The DUT did not set any error bit in the Advanced Error Reporting registers (if implemented).

If the DUT meets all these criteria, the DUT passes the test.

### **Test 52-10. RetransmitOnNak**

**ASSERTIONS COVERED:** DLL.5.2#1

The intent of this test is to ensure that a DUT retransmits a transaction for which a NAK has been issued.

**Trainer Stimulus:** link\_52-10\_RetransmitOnNak.peg

**Recording Options:** link\_layer\_common.rec

**Verification Scripts:** link\_52-10\_RetransmitOnNak.pevs

#### **Test Algorithm:**

1. Write 0x0000000F to Device Control Register, to enable Error Reporting for all error types.
2. Write 0xFFFFFFFF to Advanced Uncorrectable Error Status Register, to clear all prior error bits.
3. Write 0xFFFFFFFF to Advanced Correctable Error Status Register, to clear all prior error bits.
4. Main test stage. Do the following:
5. Send configuration read request TLP.
6. Wait for completion.
7. Send a Nak DLLP.
8. Wait for retransmitted completion.
9. Ack the retransmitted completion.
10. Read Device Status Register. No error bits should be set.
11. Read Advanced Uncorrectable Error Status Register. No error bits should be set.
12. Read Advanced Correctable Error Status Register. No error bits should be set.
13. Write 0 to Device Control Register, to disable Error Reporting for all error types.

#### **Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

#### **Verify that:**

- The CONFIG\_RD\_COMPLETION TLP for which a NAK is received is retransmitted by the DUT.



- The DUT did not set any error bit in its Device Status register and did not generate any error messages.
- The DUT did not set any error bit in the Advanced Error Reporting registers (if implemented).

If the DUT meets all these criteria, the DUT passes the test.

### Test 52-11. REPLAY\_TIMER Test

**ASSERTIONS COVERED:** DLL.5.2#1.1

The intent of this test is to ensure that a DUT REPLAY\_TIMER is working properly by not sending an ACK or NAK.

**Trainer Stimulus:** link\_52-11\_REPLAY\_TIMER.peg

**Recording Options:** link\_layer\_common.rec

**Verification Scripts:** link\_52-11\_REPLAY\_TIMER.pevs

#### Test Algorithm:

1. Write 0x0000000F to Device Control Register, to enable Error Reporting for all error types.
2. Write 0xFFFFFFFF to Advanced Uncorrectable Error Status Register, to clear all prior error bits.
3. Write 0xFFFFFFFF to Advanced Correctable Error Status Register, to clear all prior error bits.
4. Main test stage. Do the following:
5. Disable the automatic Ack/Nak.
6. Send configuration read request TLP.
7. Wait for completion.
8. Wait for the replayed completion.
9. Ack the replayed completion.
10. Read Device Status Register. Only ERR\_CORR bit should be set.
11. Read Advanced Uncorrectable Error Status Register. No error bits should be set.
12. Read Advanced Correctable Error Status Register. Only Replay Timer Timeout Status bit should be set.
13. Write 0 to Device Control Register, to disable Error Reporting for all error types.

#### Pass/Fail Criteria:

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

#### Verify that:

- The DUT retransmits the CONFIG\_RD\_COMPLETION TLP for which it received no ACK or NAK.
- The DUT set only the ERR\_CORR bit in its Device Status register and generated ERR\_CORR message.

- The DUT set “Replay Timer Timeout Status” bit in the Advanced Error Reporting Correctable Error Status register (if implemented).

If the DUT meets all these criteria, the DUT passes the test.

### Test 52-12. REPLAY\_NUM Test

**ASSERTIONS COVERED:** DLL.5.2#1.2

The intent of this test is to ensure that a DUT keeps retransmitting a transaction for which a NAK has been issued on purpose until the number of times in its REPLAY\_NUM.

**Trainer Stimulus:** link\_52-12\_REPLAY\_NUM.peg

**Recording Options:** link\_layer\_common.rec

**Verification Scripts:** link\_52-12\_REPLAY\_NUM.pevs

#### Test Algorithm:

1. Write 0x0000000F to Device Control Register, to enable Error Reporting for all error types.
2. Write 0xFFFFFFFF to Advanced Uncorrectable Error Status Register, to clear all prior error bits.
3. Write 0xFFFFFFFF to Advanced Correctable Error Status Register, to clear all prior error bits.
4. Main test stage. Do the following:
5. Set the Ack/Nak policy to “Always Nak”.
6. Send configuration read request TLP.
7. Wait for the completion TLP to arrive for REPLAY\_NUM times, while sending Nak.
8. Switch back to Automatic Ack policy.
9. Acknowledge the next completion TLP.
10. Read Device Status Register. No error bits should be set.
11. Read Advanced Uncorrectable Error Status Register. No error bits should be set.
12. Read Advanced Correctable Error Status Register. No error bits should be set.
13. Write 0 to Device Control Register, to disable Error Reporting for all error types.

#### Pass/Fail Criteria:

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

#### Verify that:

- The DUT transmits the CONFIG\_RD\_COMPLETION for REPLAY\_NUM of times with the same sequence number.
- The DUT did not set any error bit in its Device Status register and did not generate any error messages.
- The DUT did not set any error bit in the Advanced Error Reporting registers (if implemented).

If the DUT meets all these criteria, the DUT passes the test.

## Test 52-20. LinkRetrainOnRetryFail

**ASSERTIONS COVERED:** DLL.5.2#2, DLL.5.2#7

The intent of this test is to ensure that the link connected to the DUT goes into retraining after trying and failing for REPLAY\_NUM of times to get a TLP through. It also tests that the retry buffer and link states are not changed while in retraining and that the pending TLP is retransmitted after link retraining completes.

**Trainer Stimulus:** link\_52-20\_LinkRetrainOnRetryFail.peg

**Recording Options:** link\_layer\_with\_ts.rec

**Verification Scripts:** link\_52-20\_LinkRetrainOnRetryFail.pevs

### Test Algorithm:

1. Write 0x0000000F to Device Control Register, to enable Error Reporting for all error types.
2. Write 0xFFFFFFFF to Advanced Uncorrectable Error Status Register – clear all prior error bits.
3. Write 0xFFFFFFFF to Advanced Correctable Error Status Register – clear all prior error bits.
4. Main test stage. Do the following:
5. Set the Ack/Nak policy to “Disable”.
6. Send configuration read request TLP.
7. Wait (with timeout) for retraining to occur as a result of REPLAY\_NUM plus one replays.
8. Switch back to Automatic Ack policy.
9. Acknowledge the next completion TLP (that comes after link retraining).
10. Read Device Status Register. Only ERR\_CORR bit should be set.
11. Read Advanced Uncorrectable Error Status Register. No error bits should be set.
12. Read Advanced Correctable Error Status Register. Only Replay Timer Timeout and REPLAY\_NUM Rollover Status bits should be set.
13. Write 0 to Device Control Register, to disable Error Reporting for all error types.

### Pass/Fail Criteria:

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

### Verify that:

- The link did not change its state and the Config Read Completion TLP is sent after the retraining is complete (that the DATA\_BUF contains the correct data). **If not, treat it as DUT failure (DLL.5.2#7).**
- The DUT set ERR\_CORR bit in its Device Status register and generated ERR\_CORR message. **If not treat it as DUT failure (DLL.5.2#2).**
- The DUT set “REPLAY\_NUM Rollover Status” bit in the Advanced Error Reporting Correctable Error Status register (if implemented). **If not treat it as DUT failure**

**(DLL.5.2#2).**

If the DUT meets all these criteria, the DUT passes the test.

**Test 52-100. ReplayTLPOrder**

**ASSERTIONS COVERED:** DLL.5.2#10

The intent of this test is to verify that the oldest unacknowledged TLP is retransmitted first in replay followed by the other unacknowledged TLPs in the same order they were transmitted.

**Note:** Requires reconfiguration of input based on MAX\_PAYLOAD\_SIZE, link width, and other variables. You must think about configuration input.

**Trainer Stimulus:** link\_52-100\_ReplayTLPOrder.peg

**Recording Options:** link\_layer\_common.rec

**Verification Scripts:** link\_52-100\_ReplayTLPOrder.pevs

**Test Algorithm:**

1. Write 0x0000000F to Device Control Register, to enable Error Reporting for all error types.
2. Write 0xFFFFFFFF to Advanced Uncorrectable Error Status Register – clear all prior error bits.
3. Write 0xFFFFFFFF to Advanced Correctable Error Status Register – clear all prior error bits.
4. Main test stage. Do the following:
5. Disable the automatic Ack/Nak.
6. In a loop, send DUT-dependent number of Configuration Read requests with increasing addresses in the configuration space.
7. In a loop, wait for the corresponding number of completions.
8. Switch back to Automatic Ack policy.
9. Wait some time for all the outstanding completions to be acknowledged.
10. Read Device Status Register. Only ERR\_CORR bit should be set.
11. Read Advanced Uncorrectable Error Status Register. No error bits should be set.
12. Read Advanced Correctable Error Status Register. Only Replay Timer Timeout Status bit should be set.
13. Write 0 to Device Control Register, to disable Error Reporting for all error types.

**Pass/Fail Criteria:**

- Test should successfully progress though all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- a) The retransmitted CONFIG\_RD\_COMPLETION TLPs are in the original order sent.
- b) The DUT set ERR\_CORR bit in its Device Status register and generated

- ERR\_CORR message (message by itself can be replayed).
- c) The DUT set “Replay Timer Timeout Status” bit in the Advanced Error Reporting Correctable Error Status register (if implemented).

If the DUT meets all these criteria, the DUT passes the test.

### **Test 52-150. CorruptedCRC\_DLLP**

**ASSERTIONS COVERED:** DLL.5.2#15

The intent of this test is to ensure that a DUT recognizes a DLLP with a bad CRC, drops it, and logs a BAD\_DLLP port error.

**Trainer Stimulus:** link\_52-150\_CorruptedCRC\_DLLP.peg

**Recording Options:** link\_layer\_common.rec

**Verification Scripts:** link\_52-150\_CorruptedCRC\_DLLP.pevs

#### **Test Algorithm:**

1. Write 0x0000000F to Device Control Register, to enable Error Reporting for all error types.
2. Write 0xFFFFFFFF to Advanced Uncorrectable Error Status Register – clear all prior error bits.
3. Write 0xFFFFFFFF to Advanced Correctable Error Status Register – clear all prior error bits.
4. Main test stage. Do the following:
5. Disable the automatic Ack/Nak.
6. Send configuration read request TLP.
7. Wait for completion.
8. Send the Ack DLLP with bad CRC.
9. Wait for the retransmitted completion.
10. Switch back to Automatic Ack policy.
11. Read Device Status Register. Only ERR\_CORR bit should be set.
12. Read Advanced Uncorrectable Error Status Register. No error bits should be set.
13. Read Advanced Correctable Error Status Register. Only BAD\_DLLP bit should be set. (Replay Timer Timeout Status can also be set. See the Notes.)
14. Write 0 to Device Control Register, to disable Error Reporting for all error types.

#### **Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

#### **Verify that:**

- The CONFIG\_RD\_COMPLETION TLP for which the DUT received an ACK with a bad CRC is retransmitted by the DUT.
- The DUT set ERR\_CORR bit in its Device Status register and generated ERR\_CORR message.

- The DUT set “BAD DLLP” bit in the Advanced Error Reporting Correctable Error Status register (if implemented).

If the DUT meets all these criteria, the DUT passes the test.

**Note:** Because the *PETrainer™* stays in the Ack/Nak disabled mode waiting for the retransmitted completion while the ERR\_CORR message can be sent, there is a good chance the ERR\_CORR message is going to be replayed. This in turn is going to prompt another ERR\_CORR message and set a Replay Timer Timeout Status bit in the Advanced Error Reporting Correctable Error Status register (if implemented). This situation has to be tracked by the verification script and not treated as a failure.

### Test 52-160. UndefinedDLLPEncoding

**ASSERTIONS COVERED:** DLL.5.2#16

The intent of this test is to verify that the DUT silently drops any DLLP with undefined encoding (any pattern for DLLP type that is reserved right now) and no error is associated with it.

**Trainer Stimulus:** link\_52-160\_UndefinedDLLPEncoding.peg

**Recording Options:** link\_layer\_common.rec

**Verification Scripts:** link\_52-160\_UndefinedDLLPEncoding.pevs

#### Test Algorithm:

1. Write 0x0000000F to Device Control Register, to enable Error Reporting for all error types.
2. Write 0xFFFFFFFF to Advanced Uncorrectable Error Status Register, to clear all prior error bits.
3. Write 0xFFFFFFFF to Advanced Correctable Error Status Register, to clear all prior error bits.
4. Main test stage. Do the following:
  5. Disable the automatic Ack/Nak.
  6. Send configuration read request TLP.
  7. Wait for completion.
  8. Send a DLLP with undefined encoding.
  9. Wait for the retransmitted completion.
  10. Switch back to Automatic Ack policy.
  11. Read Device Status Register. Only ERR\_CORR bit should be set.
  12. Read Advanced Uncorrectable Error Status Register. No error bits should be set.
  13. Read Advanced Correctable Error Status Register. Only Replay Timer Timeout Status bit should be set.
  14. Write 0 to Device Control Register, to disable Error Reporting for all error types.

#### Pass/Fail Criteria:

- Test should successfully progress though all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- The DUT silently drops the DLLP with undefined encoding and retransmits the CONFIG\_RD\_COMPLETION TLP for which it received an undefined DLLP.
- The DUT set ERR\_CORR bit in its Device Status register and generated ERR\_CORR message.
- The DUT set “Replay Timer Timeout Status” bit in the Advanced Error Reporting Correctable Error Status register (if implemented).

If the DUT meets all these criteria, the DUT passes the test.

**Test 52-170. WrongSeqNumInAckDLLP****ASSERTIONS COVERED:** DLL.5.2#17

The intent of this test is to verify that the DUT drops any ACK DLLP that does not have a sequence number corresponding to an unacknowledged TLP and logs a BAD DLLP error associated with the port.

**Trainer Stimulus:** link\_52-170\_WrongSeqNumInAckDLLP.peg

**Recording Options:** link\_layer\_common.rec

**Verification Scripts:** link\_52-170\_WrongSeqNumInAckDLLP.pevs

**Test Algorithm:**

1. Write 0x0000000F to Device Control Register, to enable Error Reporting for all error types.
2. Write 0xFFFFFFFF to Advanced Uncorrectable Error Status Register, to clear all prior error bits.
3. Write 0xFFFFFFFF to Advanced Correctable Error Status Register, to clear all prior error bits.
4. Main test stage. Do the following:
5. Disable the automatic Ack/Nak.
6. Send configuration read request TLP.
7. Wait for completion.
8. Send an Ack DLLP with wrong sequence number (expected minus one).
9. Switch back to Automatic Ack policy.
10. Read Device Status Register. Only ERR\_CORR bit should be set.
11. Read Advanced Correctable Error Status Register. No error bits should be set.
12. Read Advanced Uncorrectable Error Status Register. Only Data Link Protocol Error bit should be set.
13. Write 0 to Device Control Register, to disable Error Reporting for all error types.

**Pass/Fail Criteria:**

- Test should successfully progress though all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- The DUT set only the ERR\_FATAL bit in its Device Status register and generated ERR\_FATAL message.
- The DUT set “BAD\_DLLP” bit in the Advanced Error Reporting Correctable Error Status register (if implemented).

If the DUT meets all these criteria, the DUT passes the test.

**Test 53-20. BadLCRC****ASSERTIONS COVERED:** DLL.5.3#2

The intent of this test is to verify that a receiver discards a TLP with a bad CRC by NAKing it and reports a BAD TLP error associated with the port.

**Trainer Stimulus:** link\_53-20\_BadLCRC.peg

**Recording Options:** link\_layer\_common.rec

**Verification Scripts:** link\_53-20\_BadLCRC.pevs

**Test Algorithm:**

1. Write 0x0000000F to Device Control Register, to enable Error Reporting for all error types.
2. Write 0xFFFFFFFF to Advanced Uncorrectable Error Status Register, to clear all prior error bits.
3. Write 0xFFFFFFFF to Advanced Correctable Error Status Register, to clear all prior error bits.
4. Main test stage. Do the following:
5. Turn off automatic sequence numbering, LCRC generation, and Replay Timer.
6. Send configuration read request TLP with bad LCRC.
7. Wait enough time for the TLP to be Nak'ed and the message to arrive.
8. Proceed with the rest of the test, setting PSNs and Tag manually.
9. Read Device Status Register. Only ERR\_CORR bit should be set.
10. Read Advanced Uncorrectable Error Status Register. No error bits should be set.
11. Read Advanced Correctable Error Status Register. Only BAD\_TLP bit should be set.
12. Write 0 to Device Control Register, to disable Error Reporting for all error types.

**Pass/Fail Criteria:**

- Test should successfully progress though all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- The DUT NAKs the TLP with bad CRC.
- The DUT set only the ERR\_CORR bit in its Device Status register and generated ERR\_CORR message.
- The DUT set “BAD\_TLP” bit in the Advanced Error Reporting Correctable Error Status register (if implemented).



If the DUT meets all these criteria, the DUT passes the test.

### Test 53-31. DuplicateTLP

**ASSERTIONS COVERED:** DLL.5.3#3.1

The intent of this test is to verify that the duplicate TLPs are handled properly by the DUT. (Duplicate TLPs have the same sequence number associated at the link layer as in the last 2048 TLPs received.)

**Trainer Stimulus:** link\_53-31\_DuplicateTLP.peg

**Recording Options:** link\_layer\_common.rec

**Verification Scripts:** link\_53-31\_DuplicateTLP.pevs

#### Test Algorithm:

1. Write 0x0000000F to Device Control Register, to enable Error Reporting for all error types.
2. Write 0xFFFFFFFF to Advanced Uncorrectable Error Status Register, to clear all prior error bits.
3. Write 0xFFFFFFFF to Advanced Correctable Error Status Register, to clear all prior error bits.
4. Main test stage. Do the following:
5. Turn off the automatic sequence numbering and Replay Timer.
6. Turn off Flow Control credits monitoring (so both TLPs can be sent no matter how many credits are advertised).
7. Send configuration read request TLP, setting the PSN manually.
8. Repeat the same read request TLP with the same PSN.
9. Wait enough time for the completion to be sent and acknowledged.
10. Turn Flow Control credits monitoring back on.
11. Proceed with the rest of the test, setting PSNs and Tag manually.
12. Read Device Status Register. No error bits should be set.
13. Read Advanced Uncorrectable Error Status Register. No error bits should be set.
14. Read Advanced Correctable Error Status Register. No error bits should be set.
15. Write 0 to Device Control Register, to disable Error Reporting for all error types.

#### Pass/Fail Criteria:

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

#### Verify that:

- The duplicated CONFIG\_RD\_REQ TLP has received two ACKs or a single coalesced ACK, but a single completion with data from the DUT.
- The DUT did not set any error bit in its Device Status register and did not generate any error messages.
- The DUT did not set any error bit in the Advanced Error Reporting registers (if

implemented).

If the DUT meets all these criteria, the DUT passes the test.

## 2.1.2 Transaction Layer Test Descriptions

### Test 2-4. TXN\_BFT\_ErrorSignaling

**ASSERTIONS COVERED:** TXN.2.15#2, TXN.2.15#3, TXN.2.15#4

The intent of this test is to verify basic signaling functionality and message generation of a slotted Endpoint device.

**Trainer Stimulus:** trans\_2-4\_TXN\_BFT\_ErrorSignaling.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** trans\_2-4\_TXN\_BFT\_ErrorSignaling.pevs

#### Test Algorithm:

1. Bring the link up.
2. Write 0x0000000F to the Device Control Register, to enable Error Reporting for all error types.
3. Send Memory Write request TLP with a wrong PSN number, which is supposed to generate a Correctable error.
4. Send a Poisoned Memory Write request TLP, which is supposed to generate a Non-Fatal error.
5. Send a Malformed Memory Write request TLP (data/length field mismatch), which is supposed to generate a Fatal error.
6. Read Device Status Register. All three error-type bits should be set.

#### Pass/Fail Criteria:

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

#### Verify that:

- Error messages of ERR\_COR, ERR\_NONFATAL, and ERR\_FATAL types are sent in proper sequence as responses to the errors created.
- Error Reporting Status bits are set in the DUT **Device Status Register** (ERR\_COR, ERR\_NONFATAL, and ERR\_FATAL are recorded in these status bits).

### Test 3-1. TXN\_BFT\_FlowControlInit

**ASSERTIONS COVERED:** TXN.6.1#14, TXN.6.1#16, TXN.5.1#1

The intent of this test is to verify that a slotted Endpoint device receiver complies with basic flow control credit advertisement requirements.

**Trainer Stimulus:** trans\_3-1\_TXN\_BFT\_FlowControlInit.peg

**Recording Options:** flow\_control\_init.rec

**Verification Scripts:** trans\_3-1\_TXN\_BFT\_FlowControlInit.pevs

**Test Algorithm:**

1. Bring the link up, capturing InitFC DLLPs.
2. Read the DUT Device Capabilities register.
3. If non-zero VCs are implemented, read the Virtual Channel Capability structure.

**Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- DUT advertises compliant flow control credit values for each supported VC, using the **Max\_Payload\_Size** value read from the Device Capabilities register.

**Test 5-1. TXN\_BFT\_VC0TCSupport**

**ASSERTIONS COVERED:** TXN.5.0#3, TXN.5.0#4

The intent of this test is to verify that slotted Endpoint devices that do not support VCs beyond the default still handle requests with non-zero TC correctly.

**Trainer Stimulus:** trans\_5-1\_TXN\_VC0TCSupport.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** trans\_5-1\_TXN\_BFT\_VC0TCSupport.pevs

**Test Algorithm:**

1. Bring the link up.
2. Send a Memory Read request for address zero with TC value of 1.
3. Wait for Completion. TC in completion should be set to 1.

**Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- The Traffic Class ID in the Completion from the DUT is the same Traffic Class ID used in the initial Request.

### 2.1.3 Tests Requiring DUT Actions

The following three tests require some DUT actions to check all assertions. When the test is executed, the PCI Express Script Automation Test Tool instructs the *PETrainer™* to wait for the specific DUT action at the proper point with the timeout of four seconds.

If the DUT does not send the expected TLP, the test proceeds to the next stage. When verification is executed, it checks for the presence of the DUT actions at the proper stages.

If the DUT did not send the TLPs expected, the corresponding assertions are not checked, and the notification is given in the log.

If no assertions could be checked for a test run, it is declared **DONE** rather than PASSED or FAILED.

### **Test 1-1. TXN\_BFT\_RequestCompletion**

**ASSERTIONS COVERED:** TXN.2.7#9, TXN.2.21#15, TXN.2.21#19, TXN.3.1#1, TXN.3.1#2

Verify Basic Request and Completion handling of slotted Endpoint devices.

**Trainer Stimulus:** trans\_1-1\_TXN\_BFT\_RequestCompletion.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** trans\_1-1\_TXN\_BFT\_RequestCompletion.pevs

#### **Test Algorithm:**

1. Bring the link up.
2. Write 0x0000000F to Device Control Register, to enable Error Reporting for all error types.
3. Write 0xFFFFFFFF to Advanced Uncorrectable Error Status Register, to clear all prior error bits.
4. Write 0xFFFFFFFF to Advanced Correctable Error Status Register, to clear all prior error bits.
5. Main test stage. Do the following:
6. Wait for the DUT to initiate a Memory Read transaction (using MemRd32 TLP). Time out in four seconds.
7. Send the Memory Read Completion with Unsupported Request status.
8. Issue a Vendor\_Defined\_Type0 request using Message TLP with vendor ID set to LeCroy vendor ID and request code of 0x7F.
9. Read Device Status Register.
10. Read Advanced Uncorrectable Error Status Register.

#### **Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

#### **Verify that:**

- a) The Completer ID field in step 2 contains the Bus and Device numbers supplied with the Configuration Write Request.
- b) The Bus and Device Numbers supplied with the Configuration Write Request to the DUT (step 2) are present in Requester ID of the Request from the DUT in step 5a. (If the DUT did not send the request, this assertion is not checked.)
- c) The following error conditions are raised and corresponding bits are set in the Device Status Register and Advanced Uncorrectable Error Status Register (if implemented):
  - Unsupported Request condition (for the Vendor request in step 5c).

- Unexpected Completion condition (only in case the DUT did not send the request in step 5a).

### Test 1-2. TXN\_BFT\_CompletionTimeout

**ASSERTIONS COVERED:** TXN.8.0#1.1, TXN.8.0#2, TXN.8.0#3, TXN.8.0#4, TXN.8.0#5

Verify Basic Completion Timer requirements of a slotted Endpoint device. The Completion Timeout timer must not expire in less than 50  $\mu$ s, but it must expire if a Request is not completed in 50 ms.

**Trainer Stimulus:** trans\_1-2\_TXN\_BFT\_CompletionTimeout.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** trans\_1-2\_TXN\_BFT\_CompletionTimeout.pevs

### Test Algorithm:

1. Bring the link up.
2. Write 0x0000000F to Device Control Register, to enable Error Reporting for all error types.
3. Write 0xFFFFFFFF to Advanced Uncorrectable Error Status Register, to clear all prior error bits.
4. Write 0xFFFFFFFF to Advanced Correctable Error Status Register, to clear all prior error bits.
5. Main test stage. Do the following:
6. Wait for the DUT to initiate a Memory Read transaction (using MemRd32 TLP). Time out in four seconds.
7. Wait for 49 microseconds.
8. Send the Memory Read Completion with Unsupported Request status.
9. Read the DUT Device Status Register and Advanced Uncorrectable Error Status Register.
10. Wait for the DUT to initiate another Memory Read transaction (using MemRd32 TLP). Time out in four seconds.
11. Wait for 50 milliseconds plus 1 microsecond.
12. Send the Memory Read Completion with Unsupported Request status.
13. Read Device Status Register.
14. Read Advanced Uncorrectable Error Status Register.

### Pass/Fail Criteria:

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

### Verify that:

- The following error conditions are set by the DUT in case the DUT has initiated the memory requests (if it did not, no assertions can be checked, and the test is declared **DONE**):
  - No errors are set in error registers in step 5d.

- The ERR\_CORR bit is set in the Device Status register in step 6.
- The Completion Time-out bit is set in the Advanced Uncorrectable Error Status register (if implemented) in step 7.

### Test 2-1. TXN\_BFT\_LegacyInt

**ASSERTIONS COVERED:** TXN.2.13#3, TXN.2.13#4, TXN.2.13#7, TXN.2.13#10, TXN.2.13#11, TXN.2.13#13, TXN.2.13#14, TXN.2.13#21

Verify Basic INTx message support requirements of slotted Endpoint devices.

**Trainer Stimulus:** trans\_2-1\_TXN\_BFT\_LegacyInt.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** trans\_2-1\_TXN\_BFT\_LegacyInt.pevs

#### Test Algorithm:

1. Bring the link up.
2. Read the DUT Interrupt Pin Register value (register address 0x3C).
3. Read the DUT Interrupt Disable bit value (register address 0x4).
4. Main test stage. Do the following:
  5. Wait for the DUT to assert interrupt (using ASSERT\_INTx Message TLP). Time out in four seconds.
  6. Set the Interrupt Disable bit.
  7. Wait enough time for DUT to de-assert the asserted interrupt.

#### Pass/Fail Criteria:

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

#### Verify that:

- In case the DUT asserted interrupt (if it did not, the results are not applicable, and the test is marked DONE):
  - The DUT has asserted the interrupt using the appropriately mapped Message TLP.
  - The DUT has asserted the interrupt in step 4c using the appropriately mapped Message TLP.

### 2.1.4 Root Complex Tests

In the following test descriptions, the term “Issue command to the Driver” is often used. This term indicates that the following sequence of steps is performed.

1. Write the Command DWORD to the Command system buffer.
2. Assert the legacy interrupt to interrupt the system.
3. Wait for the Memory Write transaction from the system that acknowledges the interrupt.
4. Deassert the legacy interrupt.

**Note:** All the Root Complex tests start with re-training and initialization of the PCI Express link, so this first default step is omitted in the test descriptions.

## 2.1.5 Link Layer Test Descriptions

### Test 41-20. ReservedFieldsDLLPReceive

**ASSERTIONS COVERED:** DLL.4.1#2

The intent of this test is to verify that the DUT truly ignores reserved fields in an ACK DLLP by sending arbitrary data in those fields.

**Trainer Stimulus:** link\_41-20\_ReservedFieldsDLLPReceive.peg

**Recording Options:** link\_layer\_filter\_ts.rec

**Verification Scripts:** link\_41-20\_ReservedFieldsDLLPReceive.pevs

#### Test Algorithm:

1. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.
2. Main test stage. Do the following:
3. Switch to the 'Disable' ACK/NAK policy.
4. Re-train and initialize the link.
5. Wait for the Set\_Slot\_Power\_Limit Message from the Root.
6. Send an Ack DLLP with non-zero reserved field.
7. Switch back to the 'Automatic' ACK/NAK policy.
8. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).

#### Pass/Fail Criteria:

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

#### Verify that:

- a) The DUT ignored the non-zero field(s) in the ACK and did not retransmit the Set\_Slot\_Power\_Limit Message TLP.
- b) The DUT did not set any error bits in its Device Status register.
- c) The DUT did not set any error bits in the Advanced Error Reporting registers (if implemented). This includes the Advanced Uncorrectable Error Status register, Advanced Correctable Error Status register, and Root Error Status register.

If the DUT meets all these criteria, the DUT passes the test.

### Test 52-10. RetransmitOnNak

**ASSERTIONS COVERED:** DLL.5.2#1

The intent of this test is to ensure that a DUT retransmits a transaction for which a NAK has been issued.

**Trainer Stimulus:** link\_52-10\_RetransmitOnNak.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** link\_52-10\_RetransmitOnNak.pevs

**Test Algorithm:**

1. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.
2. Main test stage. Do the following:
3. Issue command to the Driver to perform 1-DWORD Memory Write to the Device Emulator.
4. Switch to the 'Always Nak' ACK/NAK policy.
5. Wait for the Memory Write TLP.
6. Switch back to the 'Automatic' ACK/NAK policy.
7. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).

**Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- The Memory Write TLP for which a NAK is received is retransmitted by the DUT.
- The DUT did not set any error bits in its Device Status register.
- The DUT did not set any error bits in the Advanced Error Reporting registers (if implemented). This includes Advanced Uncorrectable Error Status register, Advanced Correctable Error Status register and Root Error Status register.

If the DUT meets all these criteria, the DUT passes the test.

**Test 52-11. REPLAY\_TIMER Test**

**ASSERTIONS COVERED:** DLL.5.2#1.1

The intent of this test is to ensure that a DUT REPLAY\_TIMER is working properly by not sending an ACK or NAK.

**Trainer Stimulus:** link\_52-11\_REPLAY\_TIMER.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** link\_52-11\_REPLAY\_TIMER.pevs

**Test Algorithm:**

1. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.



2. Main test stage. Do the following:
3. Issue command to the Driver to perform 1-DWORD Memory Write to the Device Emulator.
4. Switch to the 'Disable' ACK/NAK policy.
5. Wait for the Memory Write TLP.
6. Switch back to the 'Automatic' ACK/NAK policy.
7. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).

**Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- The DUT retransmits the Memory Write TLP for which it received no ACK or NAK.
- The DUT set only the ERR\_CORR bit in its Device Status register.
- The DUT did not set any error bits in the Advanced Uncorrectable Error Status register (if implemented).
- The DUT set the "Replay Timer Timeout Status" bit in the Advanced Error Reporting Correctable Error Status register (if implemented).
- The DUT set the "Correctable Error Received" bit in the Root Error Status register (if implemented and applicable).

If the DUT meets all these criteria, the DUT passes the test.

**Test 52-12. REPLAY\_NUM Test****ASSERTIONS COVERED:** DLL.5.2#1.2

The intent of this test is to ensure that a DUT keeps retransmitting a transaction for which a NAK has been issued on purpose until the number of times in its REPLAY\_NUM.

**Trainer Stimulus:** link\_52-12\_REPLAY\_NUM.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** link\_52-12\_REPLAY\_NUM.pevs

**Test Algorithm:**

1. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.
2. Main test stage. Do the following:
3. Issue command to the Driver to perform 1-DWORD Memory Write to the Device Emulator.
4. Switch to the 'Disable' ACK/NAK policy.
5. Wait for the Memory Write TLP to be sent REPLAY\_NUM times.
6. Switch back to the 'Automatic' ACK/NAK policy.

7. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).

**Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- The DUT transmits the Memory Write TLP for REPLAY\_NUM of times with the same sequence number.
- The DUT did not set any error bits in its Device Status register.
- The DUT did not set any error bits in the Advanced Error Reporting registers (if implemented). This includes the Advanced Uncorrectable Error Status register, Advanced Correctable Error Status register, and Root Error Status register.

If the DUT meets all these criteria, the DUT passes the test.

**Test 52-20. LinkRetrainOnRetryFail**

**ASSERTIONS COVERED:** DLL.5.2#2, DLL.5.2#7

The intent of this test is to ensure that the link connected to the DUT goes into retraining after trying and failing for REPLAY\_NUM of times to get a TLP through. It also tests that the retry buffer and link states are not changed while in retraining and that the pending TLP is retransmitted after link retraining completes.

**Trainer Stimulus:** link\_52-20\_LinkRetrainOnRetryFail.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** link\_52-20\_LinkRetrainOnRetryFail.pevs

**Test Algorithm:**

1. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.
2. Main test stage. Do the following:
3. Issue command to the Driver to perform 1-DWORD Memory Write to the Device Emulator.
4. Switch to the 'Disable' ACK/NAK policy.
5. Wait (with timeout) for the link to be retrained as a result of Memory Write TLP to be sent REPLAY\_NUM + 1 times.
6. Switch back to the 'Automatic' ACK/NAK policy.
7. Acknowledge the TLP that comes after link retraining.
8. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).

**Pass/Fail Criteria:**

- Test should successfully progress through all test stages.

- All test stages should be executed without protocol violations.

**Verify that:**

- The link did not change its state, and the Memory Write TLP is sent after the retraining is complete (that the DATA\_BUF contains the correct data). **If not, treat it as DUT failure (DLL.5.2#7).**
- The DUT set the ERR\_CORR bit in its Device Status register. **If not, treat it as DUT failure (DLL.5.2#2).**
- The DUT did not set any error bits in the Advanced Uncorrectable Error Status register (if implemented).
- The DUT set the "REPLAY\_NUM Rollover Status" bit in the Advanced Error Reporting Correctable Error Status register (if implemented). **If not, treat it as DUT failure (DLL.5.2#2).**
- The DUT set the "Correctable Error Received" bit in the Root Error Status register (if implemented and applicable).

If the DUT meets all these criteria, the DUT passes the test.

**Test 52-100. ReplayTLPOrder**

**ASSERTIONS COVERED:** DLL.5.2#10

The intent of this test is to verify that the oldest unacknowledged TLP is retransmitted first in replay, followed by the other unacknowledged TLPs in the same order that they were transmitted.

**Trainer Stimulus:** link\_52-100\_ReplayTLPOrder.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** link\_52-100\_ReplayTLPOrder.pevs

**Test Algorithm:**

1. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.
2. Main test stage. Do the following:
3. Issue command to the Driver to perform a series of six 1-DWORD Memory Writes to the Device Emulator with zero delay.
4. Switch to the 'Disable' ACK/NAK policy.
5. Wait for the Memory Write TLP to be sent six times.
6. Switch back to the 'Automatic' ACK/NAK policy.
7. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).

**Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

**Verify that**

- The retransmitted Memory Write TLPs are in the original order sent.
- The DUT set the ERR\_CORR bit in its Device Status register.
- The DUT did not set any error bits in the Advanced Uncorrectable Error Status register (if implemented).
- The DUT set the “Replay Timer Timeout Status” bit in the Advanced Error Reporting Correctable Error Status register (if implemented).
- The DUT set the “Correctable Error Received” bit in the Root Error Status register (if implemented and applicable).

If the DUT meets all these criteria, the DUT passes the test.

**Test 52-150. CorruptedCRC\_DLLP**

**ASSERTIONS COVERED:** DLL.5.2#15

The intent of this test is to ensure that a DUT recognizes a DLLP with a bad CRC, drops it, and logs a BAD\_DLLP port error.

**Trainer Stimulus:** link\_52-150\_CorruptedCRC\_DLLP.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** link\_52-150\_CorruptedCRC\_DLLP.pevs

**Test Algorithm**

1. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.
2. Main test stage. Do the following:
3. Issue command to the Driver to perform 1-DWORD Memory Write to the Device Emulator.
4. Switch to the 'Disable' ACK/NAK policy.
5. Wait for the Memory Write TLP.
6. Send the Ack DLLP with bad CRC.
7. Switch back to the 'Automatic' ACK/NAK policy.
8. Acknowledge the retransmitted TLP.
9. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).

**Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- The Memory Write TLP for which the DUT received an ACK with a bad CRC is retransmitted by the DUT.
- The DUT set the ERR\_CORR bit in its Device Status register.
- The DUT did not set any error bits in the Advanced Uncorrectable Error Status

- register (if implemented).
- The DUT set the “BAD DLLP” and “Replay Timer Timeout Status” bits in the Advanced Error Reporting Correctable Error Status register (if implemented).
- The DUT set the “Correctable Error Received” and “Multiple Correctable Errors Received” bits in the Root Error Status register (if implemented and applicable).

If the DUT meets all these criteria, the DUT passes the test.

### **Test 52-160. UndefinedDLLPEncoding**

**ASSERTIONS COVERED:** DLL.5.2#16

The intent of this test is to verify that the DUT silently drops any DLLP with undefined encoding (any pattern for DLLP type that is currently reserved) and no error is associated with it.

**Trainer Stimulus:** link\_52-160\_UndefinedDLLPEncoding.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** link\_52-160\_UndefinedDLLPEncoding.pevs

### **Test Algorithm**

1. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.
2. Main test stage. Do the following:
3. Issue command to the Driver to perform 1-DWORD Memory Write to the Device Emulator.
4. Switch to the 'Disable' ACK/NAK policy.
5. Wait for the Memory Write TLP.
6. Send a DLLP with undefined encoding.
7. Switch back to the 'Automatic' ACK/NAK policy.
8. Acknowledge the retransmitted TLP.
9. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).

### **Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

### **Verify that**

- The DUT silently drops the DLLP with undefined encoding and retransmits the Memory Write TLP for which it received an undefined DLLP.
- The DUT set the ERR\_CORR bit in its Device Status register.
- The DUT did not set any error bits in the Advanced Uncorrectable Error Status register (if implemented).
- The DUT set the “Replay Timer Timeout Status” bit in the Advanced Error Reporting Correctable Error Status register (if implemented).

- The DUT set the “Correctable Error Received” bit in the Root Error Status register (if implemented and applicable).

If the DUT meets all these criteria, the DUT passes the test.

### **Test 52-170. WrongSeqNumInAckDLLP**

**ASSERTIONS COVERED:** DLL.5.2#17

The intent of this test is to verify that the DUT drops any ACK DLLP that does not have a sequence number corresponding to an unacknowledged TLP and logs a BAD DLLP error associated with the port.

**Trainer Stimulus:** link\_52-170\_WrongSeqNumInAckDLLP.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** link\_52-170\_WrongSeqNumInAckDLLP.pevs

### **Test Algorithm**

1. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.
2. Main test stage. Do the following:
3. Issue command to the Driver to perform 1-DWORD Memory Write to the Device Emulator.
4. Switch to the 'Disable' ACK/NAK policy.
5. Wait for the Memory Write TLP.
6. Send an Ack DLLP with wrong sequence number.
7. Switch back to the 'Automatic' ACK/NAK policy.
8. Acknowledge the retransmitted TLP.
9. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).

### **Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

### **Verify that:**

- The DUT set the ERR\_CORR and ERR\_FATAL bits in its Device Status register.
- The DUT set the “Data Link Error” bit in the Advanced Error Reporting Uncorrectable Error Status register (if implemented).
- The DUT set the “Replay Timer Timeout Status” bit in the Advanced Error Reporting Correctable Error Status register (if implemented).
- The DUT set the “Correctable Error Received”, “Uncorrectable Error Received”, “Fatal Error Received”, and “First Uncorrectable Fatal” bits in the Root Error Status register (if implemented and applicable).

If the DUT meets all these criteria, the DUT passes the test.

### Test 53-20. BadLCRC

**ASSERTIONS COVERED:** DLL.5.3#2

The intent of this test is to verify that a receiver discards a TLP with a bad CRC by NAKing it and reports a BAD TLP error associated with the port.

**Trainer Stimulus:** link\_53-20\_BadLCRC.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** link\_53-20\_BadLCRC.pevs

**Test Algorithm:**

1. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.
2. Main test stage. Do the following:
3. Turn off automatic LCRC generation and automatic sequence numbering.
4. Send a Memory Read Request TLP with bad LCRC.
5. Wait for some time.
6. Send the same Memory Read Request TLP, now with good LCRC.
7. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Memory Writes).
8. Switch back to automatic settings, retrain, and initialize the link.

**Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- The DUT NAKs the TLP with bad CRC.
- The DUT completes the Memory Read transaction.
- The DUT set only the ERR\_CORR bit in its Device Status register.
- The DUT did not set any error bits in the Advanced Uncorrectable Error Status register (if implemented).
- The DUT set the "BAD\_TLP" bit in the Advanced Error Reporting Correctable Error Status register (if implemented).
- The DUT set the "Correctable Error Received" bit in the Root Error Status register (if implemented and applicable).

If the DUT meets all these criteria, the DUT passes the test.

### Test 53-31. DuplicateTLP

**ASSERTIONS COVERED:** DLL.5.3#3.1

The intent of this test is to verify that the duplicate TLPs are handled properly by the DUT. (Duplicate TLPs have the same sequence number associated at the link layer as in the latest 2048 TLPs received.)

**Trainer Stimulus:** link\_53-31\_DuplicateTLP.peg

**Recording Options:** link\_layer\_common.rec

**Verification Scripts:** link\_53-31\_DuplicateTLP.pevs

**Test Algorithm:**

1. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.
2. Main test stage. Do the following:
3. Turn off automatic LCRC generation and automatic sequence numbering.
4. Send a Memory Read Request TLP.
5. Immediately send the same Memory Read Request TLP, with the same PSN.
6. Wait enough time for the DUT to complete the transaction.
7. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Memory Writes).
8. Switch back to automatic settings, retrain, and initialize the link.

**Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- The duplicated Memory Read Request TLP has received two ACKs or a single coalesced ACK and a single completion with data from the DUT.
- The DUT did not set any error bits in its Device Status register.
- The DUT did not set any error bits in the Advanced Error Reporting registers (if implemented). This includes the Advanced Uncorrectable Error Status register, Advanced Correctable Error Status register, and Root Error Status register.

If the DUT meets all these criteria, the DUT passes the test.

## 2.1.6 Transaction Layer Test Descriptions

### Test 1-1. TXN\_BFT\_RequestCompletion

**ASSERTIONS COVERED:** TXN.2.7#9, TXN.2.21#15, TXN.2.21#19, TXN.3.1#1, TXN.3.1#2

Verify Basic Request and Completion handling of Root Complex devices.

**Trainer Stimulus:** trans\_1-1\_TXN\_BFT\_RequestCompletion.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** trans\_1-1\_TXN\_BFT\_RequestCompletion.pevs

**Test Algorithm:**

1. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.



2. Issue command to the Driver to perform a Tag synchronization sequence (if needed) to ensure the next transaction is done with an expected Tag value.
3. Issue command to the Driver to perform a Memory Read transaction.
4. Wait for the Memory Read Request and issue Memory Read Completion with Unsupported Request status.
5. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).
6. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.
7. Perform Memory Read transaction from the DUT (command memory system area).
8. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).

**Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- The DUT system constructed Memory Read Request and Completion TLPs properly.
- The DUT did not set any error bits in its Device Status register.
- The DUT did not set any error bits in the Advanced Error Reporting registers (if implemented). This includes the Advanced Uncorrectable Error Status register, Advanced Correctable Error Status register, and Root Error Status register.

If the DUT meets all these criteria, the DUT passes the test.

**Test 1-2. TXN\_BFT\_CompletionTimeout**

**ASSERTIONS COVERED:** TXN.8.0#1.1, TXN.8.0#2, TXN.8.0#3, TXN.8.0#4, TXN.8.0#5

Verify Basic Completion Timer requirements of a Root Complex device. The Completion Timeout timer must not expire in less than 50  $\mu$ s, but it must expire if a Request is not completed in 50 ms.

**Trainer Stimulus:** trans\_1-2\_TXN\_BFT\_CompletionTimeout.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** trans\_1-2\_TXN\_BFT\_CompletionTimeout.pevs

**Test Algorithm:**

1. Issue command to the Driver to read and reflect to the link the value of the Uncorrectable Error Severity register for the Root or Switch port to which the Device Emulator is attached. The Verification script uses this value to determine how Completion Timeout and Unexpected Completion errors should be reported.

2. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.
3. Issue command to the Driver to perform a Tag synchronization sequence (if needed) to ensure the next transaction is done with an expected Tag value.
4. Issue command to the Driver to perform a Memory Read transaction.
5. Wait for the Memory Read Request, then wait for 49 microseconds and issue Memory Read Completion.
6. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).
7. Issue command to the Driver to perform a Tag synchronization sequence (if needed) to ensure the next transaction is done with an expected Tag value.
8. Issue command to the Driver to perform a Memory Read transaction.
9. Wait for the Memory Read Request, then wait for 50 milliseconds plus 1 microsecond and issue Memory Read Completion.
10. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).

**Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- The following error conditions are set by the DUT:
  - No errors are set in error registers in step 6.
  - The ERR\_NONFATAL and/or ERR\_FATAL bits are set in the Device Status register in step 10, depending on the severity of the Completion Timeout and Unexpected Completion errors.
  - The Completion Timeout and Unexpected Completion error bits are set in the Advanced Uncorrectable Error Status register (if implemented) in step 10.
  - No errors are set in the Advanced Correctable Error Status register (if implemented) in step 10.
  - Error bits are set in the Root Error Status register (if implemented) in step 10, depending on the severity of the Completion Timeout and Unexpected Completion errors.
- If the DUT meets all these criteria, the DUT passes the test.

**Test 2-4. TXN\_BFT\_ErrorSignaling**

**ASSERTIONS COVERED:** TXN.2.15#2, TXN.2.15#3, TXN.2.15#4

The intent of this test is to verify basic error signaling functionality of a Root or Switch port related to Uncorrectable (Non-Fatal or Fatal) errors.

**Trainer Stimulus:** trans\_2-4\_TXN\_BFT\_ErrorSignaling.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** trans\_2-4\_TXN\_BFT\_ErrorSignaling.pevs

**Test Algorithm:**

1. Issue command to the Driver to read and reflect to the link the value of the Uncorrectable Error Severity register for the Root or Switch port to which the Device Emulator is attached. The Verification script uses this value to determine how each of the errors created during the test should be reported.
2. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.
3. Send Memory Write TLP with Length = 1, but no Payload, thus creating a Malformed TLP error.
4. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).
5. Issue four commands to the Driver to read and reflect to the link the value of the four DWORDS of the Header Log register for the Root or Switch port to which the Device Emulator is attached. (If Advanced Error Reporting capability is not implemented, the values are ignored).
6. Issue command to the Driver to clear all error status bits for the Root or Switch port to which the Device Emulator is attached.
7. Send Memory Read Completion that is not expected, thus creating an Unexpected Completion error.
8. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).
9. Issue four commands to the Driver to read and reflect to the link the value of the four DWORDS of the Header Log register for the Root or Switch port to which the Device Emulator is attached.
10. Send Memory Write TLP with EP = 1, thus creating a Poisoned TLP error.
11. Issue command to the Driver to reflect the current values of the Error Reporting Registers for the port to which the Device Emulator is attached (using Configuration Writes).
12. Issue four commands to the Driver to read and reflect to the link the value of the four DWORDS of the Header Log register for the Root or Switch port to which the Device Emulator is attached.

**Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- After each of the three error types, the DUT set the corresponding Non-Fatal or Fatal bit in the Device Status register for the Root/Switch port.

- The DUT did not set any error bits in the Advanced Correctable Error Status register (if implemented), for all three types of errors.
- The DUT set the corresponding error bits in the Advanced Uncorrectable Error Status register (if implemented), for each of the three types of errors.
- The DUT logged the four DWORDS of the header for each of the three erroneous TLPs the Device Emulator generated in the Header Log register (if implemented).
- The DUT set the corresponding bits in the Root Error Status register (if implemented), for each of the three types of errors.
- If the DUT meets all these criteria, the DUT passes the test.

### Test 3-1. TXN\_BFT\_FlowControlInit

Assertions covered: TXN.6.1#14, TXN.6.1#16, TXN.5.1#1

The intent of this test is to verify that a Root Complex device receiver complies with basic flow control credit advertisement requirements.

**Trainer Stimulus:** trans\_3-1\_TXN\_BFT\_FlowControlInit.peg

**Recording Options:** link\_layer\_init\_fc.rec

**Verification Scripts:** trans\_3-1\_TXN\_BFT\_FlowControlInit.pevs

Test Algorithm:

Bring the link up, capturing InitFC DLLPs and the Set\_Slot\_Power\_Limit message from the Root.

Pass/Fail Criteria:

Test should successfully progress through all test stages.

All test stages should be executed without protocol violations.

**Verify that:**

- The DUT advertises compliant flow control credit values for each supported VC, using the Max\_Payload\_Size value in the Device Capabilities register of the Root or Switch port.
- The DUT sends the proper Set\_Slot\_Power\_Limit message on the Link Up event, according to the values read from the Device Capabilities register of the Root or Switch port.
- If the DUT meets all these criteria, the DUT passes the test.

### Test 4-0. TXN\_BFT\_TransactionTypes

**ASSERTIONS COVERED:** N/A

Verify Root Complex capability to create transactions of various types and pass data between Root Complex and an Endpoint.

**Trainer Stimulus:** trans\_4-0\_TXN\_BFT\_TransactionTypes.peg

**Recording Options:** link\_layer.rec

**Verification Scripts:** trans\_4-0\_TXN\_BFT\_TransactionTypes.pevs

**Test Algorithm:**

1. Issue command to the Driver to perform a Tag synchronization sequence (if needed) to ensure the next transaction is done with an expected Tag value.
2. Issue command to the Driver to perform a Memory Read transaction.
3. Wait for the Memory Read Request and issue Memory Read Completion with some data.
4. Issue command to the Driver to perform a Tag synchronization sequence (if needed) to ensure the next transaction is done with an expected Tag value.
5. Issue command to the Driver to perform a 1-byte IO Read transaction.
6. Wait for the IO Read Request and issue IO Read Completion with certain data. The Driver places this data in the system buffer.
7. Issue command to the Driver to perform a Tag synchronization sequence (if needed) to ensure the next transaction is done with an expected Tag value.
8. Issue command to the Driver to perform a 1-byte IO Write transaction with the data that was placed in the system buffer.
9. Issue command to the Driver to perform a 1-byte Configuration Read transaction from a certain location in the Device Emulator configuration space. The Driver places this data in the system buffer.
10. Issue command to the Driver to perform a 1-byte Configuration Write transaction with the data that was placed in the system buffer.

**Pass/Fail Criteria:**

- Test should successfully progress through all test stages.
- All test stages should be executed without protocol violations.

**Verify that:**

- a) The DUT has properly constructed and executed all the types of transactions in the test.
- b) The data used for the IO Write was the one read by the IO Read.
- c) The data used for the Configuration Write was the one read by the Configuration Read.

If the DUT meets all these criteria, the DUT passes the test.



# Chapter 3

## Gen3 Compliance Testing

---

### 3.1 Hardware Requirements

This test procedure should be followed for Link and Transaction Layer testing of PCI Express 3.0 Add-In cards using the Teledyne LeCroy Summit T3, Summit Z3 and the Z3 Test Platform.

In order to run the tests, Summit Z3 must be licensed for Gen3 Compliance Test Suite.

Testing will be carried out at:

- 8GT/s, 5GT/s and 2.5GT/s for cards capable of 8GT/s.
- 5GT/s and 2.5GT/s for cards capable of 5GT/s.
- 2.5GT/s for devices that support 2.5GT/s only.

### 3.2 Software Requirements

The Summit T3 and Summit Z3 software required for running the Link and Transaction layer tests on the Teledyne LeCroy T3 and Z3 can be downloaded from the Teledyne LeCroy website:

<http://teledynelecroy.com/support/softwaredownload/documents.aspx>

Use PETracer 7.15 or newer software versions to run the Gen3 Compliance Test Suite.

### 3.3 Software Installation

The testing is carried out using the Script Automation Test Tool which is installed when the main PETracer GUI is installed.

### 3.4 Hardware Installation

The following components required for testing:

- Summit Z3-16 Exerciser
- Gen3-capable Summit Analyzer
- Summit Z3 Test Platform
- I-Pass Y cable
- 2x USB cables
- 1 Controller PC

1. Plug the Summit Z3-16 into the slot marked on the test platform.
2. Plug the DUT into the slot marked DUT.
3. Connect the I-Pass cable between the test platform and the analyzer, with connection A on the Upstream side and connection B on the downstream side.
4. Connect the USB cables from the Z3-16 and the T3-8 or T3-16 to an available port on the controller PC.

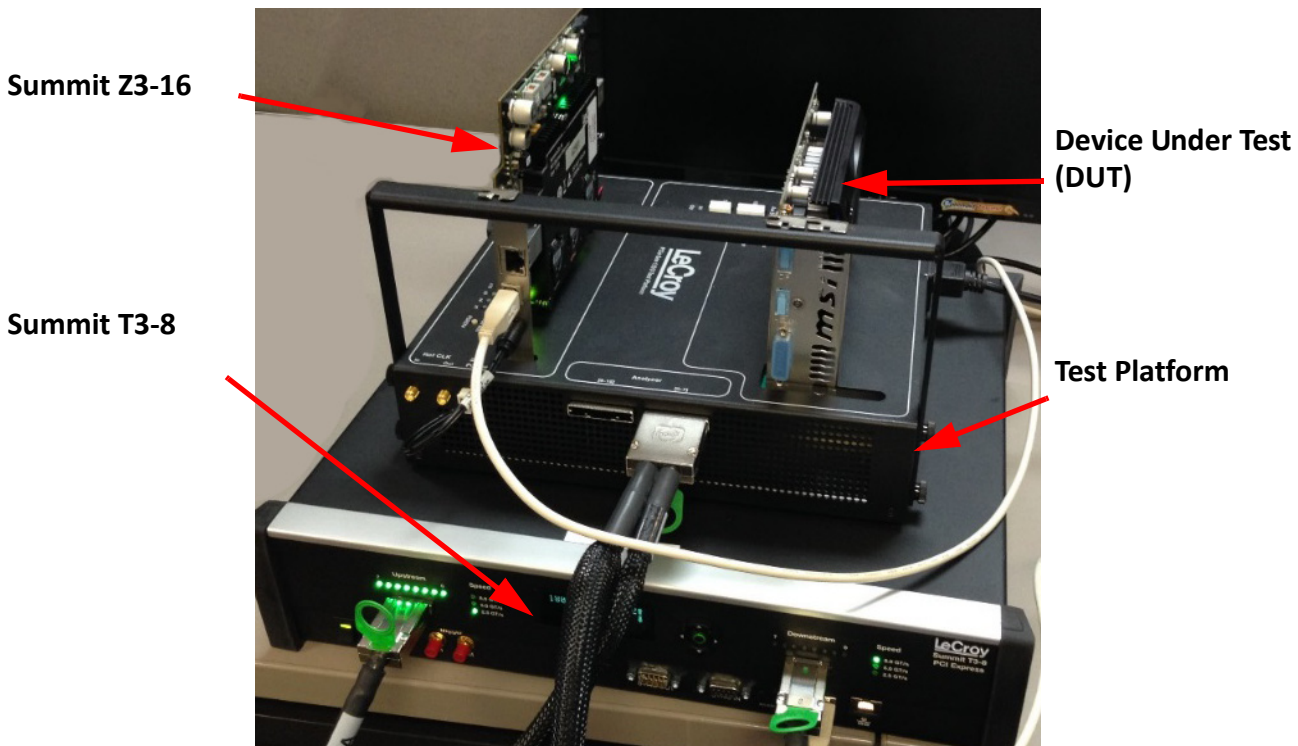


Figure 3.1: Hardware setup.

### 3.5 Using the Script Automation Test Tool

1. The test cases are implemented in the Teledyne LeCroy Script Automation Test Tool. This is launched from the following location:
  - Start Menu -> All Programs -> LeCroy -> PETracer -> LeCroy PCI Express Script Automation Test Tool
  - When the software begins to run, the following window will appear ([Figure 3.2](#)). All testing is carried out using this GUI.
2. Ensure that the test platform DUT power is turned on using the switch on the top of the test platform.



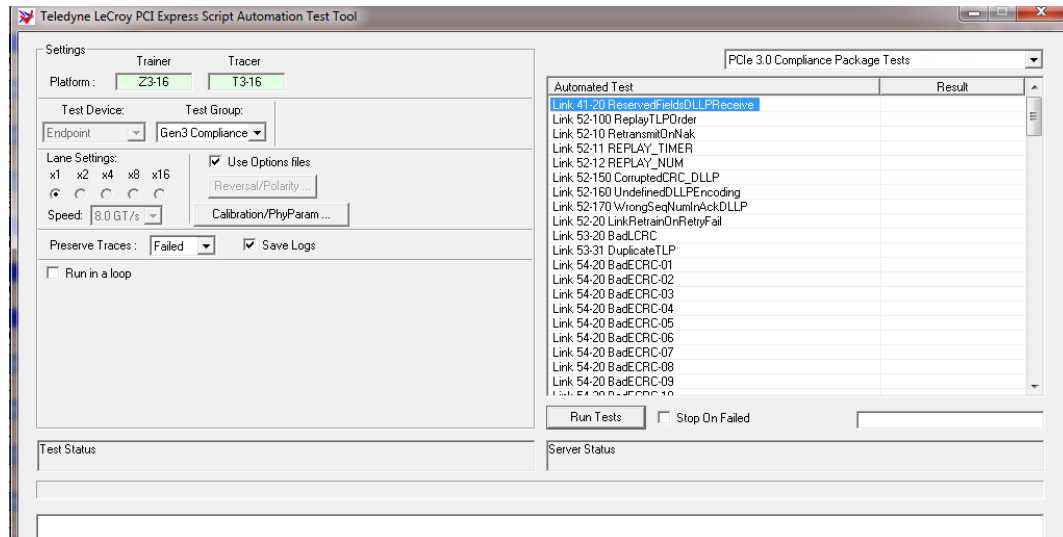


Figure 3.2: Script Automation Test Tool GUI.

- There is a drop-down list in the top right that by default shows Other tests. Change this selection to **PCIe 3.0 Compliance Package Tests**.

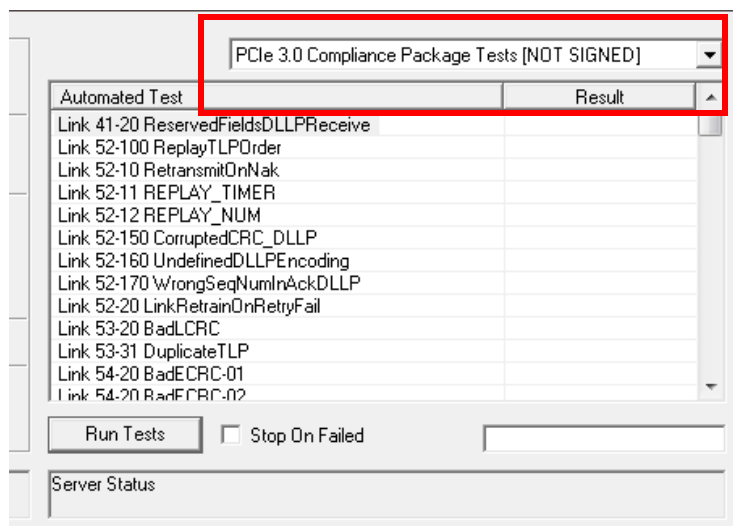


Figure 3.3: PCIe 3.0 Compliance Package Tests selected.

4. Make sure that the software has detected both the Z3 and T3-8 or T3-16 hardware; this is shown in the **Settings** box ([Figure 3.4](#)).

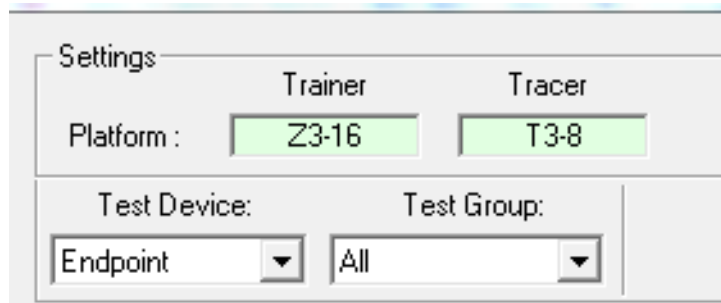


Figure 3.4: Hardware connection and test group selection.

- If the Trainer and Tracer boxes do not indicate that the hardware is found, then refer to the user manual to ensure that the drivers for the devices are correctly installed and that the hardware is connected to the controller PC.
5. Choose which tests to run and make sure that all settings are correct.

**Test Device:** The testing defined here is for add-in cards or switch upstream ports, so Test Device should be selected as "Endpoint".

**Test Group:** The tests are organized in groups. Not all tests included are required for official compliance; some tests are included for information only. Tests in the GEN3 group are the tests that are required for PCI Express 3.0 official testing.

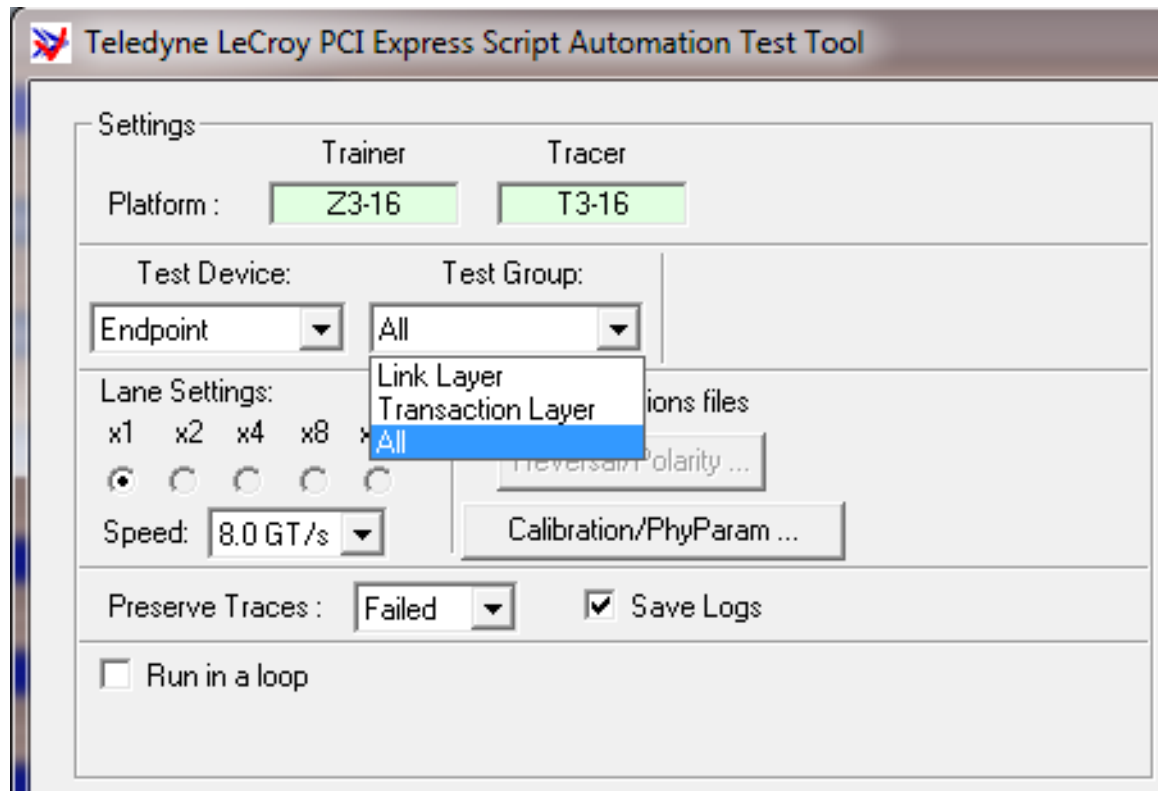


Figure 3.5: Link width and speed selection.

**Lane Settings:** Carry out all testing at x1 link width. See [Figure 3.5](#).

**Speed:** Set the speed to 8GT/s for testing 8GT/s capable and 5GT/s capable devices. Set the speed to either 2.5GT/s or 8GT/s for 2.5GT/s devices. The individual test cases will determine the actual speed of the link.

**Preserve Traces:** When each test case is run, you can choose whether the created trace files will be stored every time or only when a test case fails. This does not change the results of the test.

- Note that the resulting log files and trace files can be quite large if the option to save all traces is chosen.

**Calibration/PhyParam:** If the link needs different calibration than the default, you can use the **Calibration/PhyParameters** button to browse to the location of your calibration file. See [Figure 3.6](#).

- Perform calibration from PETracer SW and save the probe settings as a \*.ps file. Then call this file from the Script Automation Test Tool.

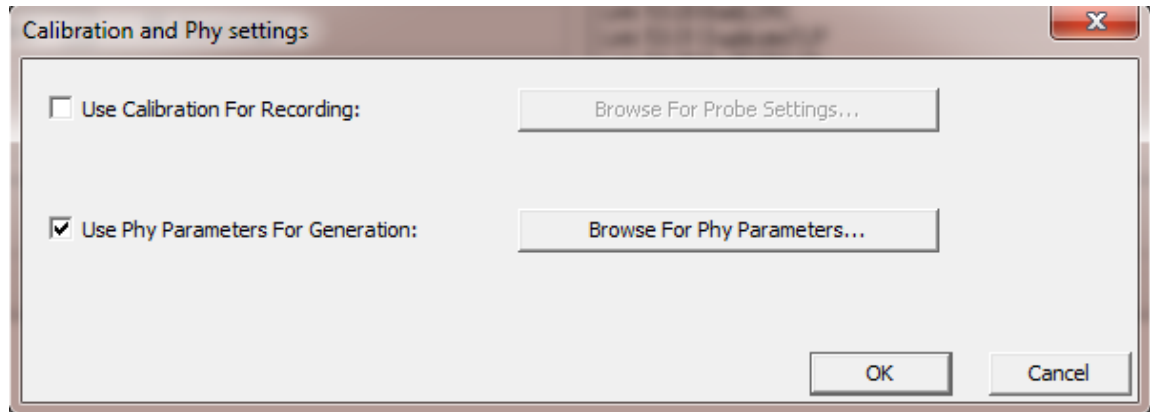


Figure 3.6: Calibration selection.

6. Select the desired test cases from the list. Multiple test cases can be selected and run by holding down the "CTRL" key when selecting tests.
7. After selecting the desired tests, select the **Run Tests** button to execute the test cases. See [Figure 3.7](#).

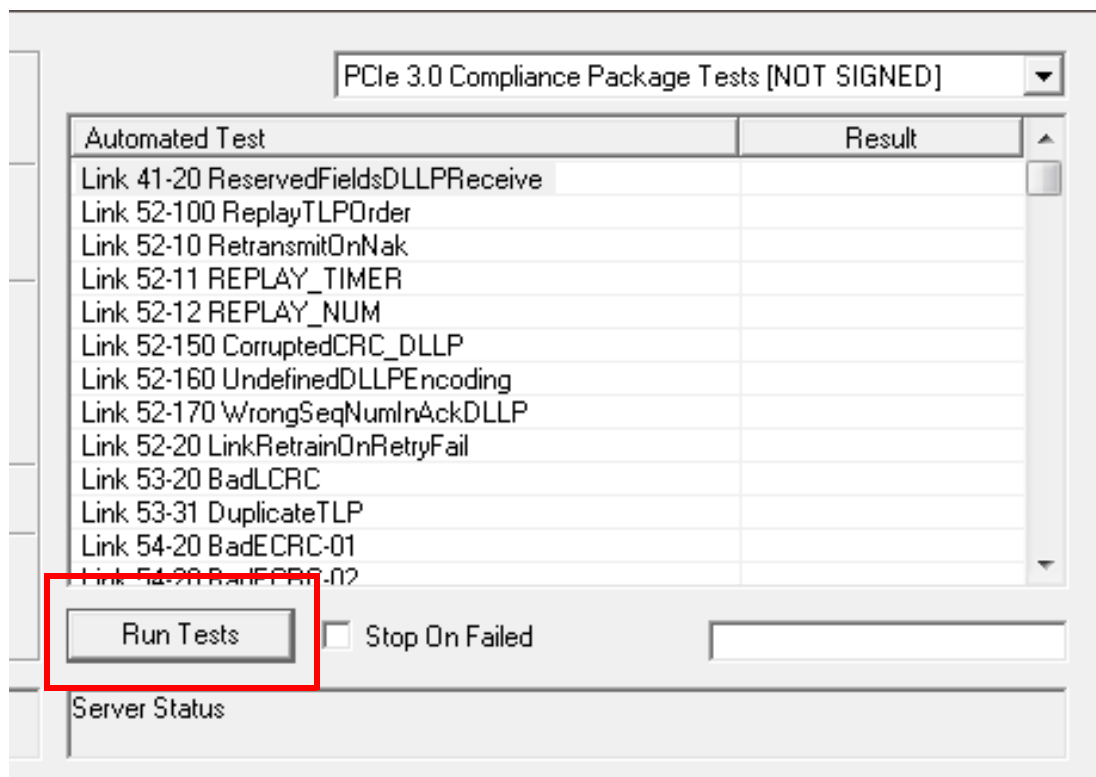


Figure 3.7: Test Selection.

8. The test results are shown in the **Result** column in the GUI (Figure 3.8). The test logs will be shown on the lower section and will also be written to log files in the following directory:

C:\Users\Public\Documents\LeCroy\PETracer\ScriptAutomationTestTool\TestLog\

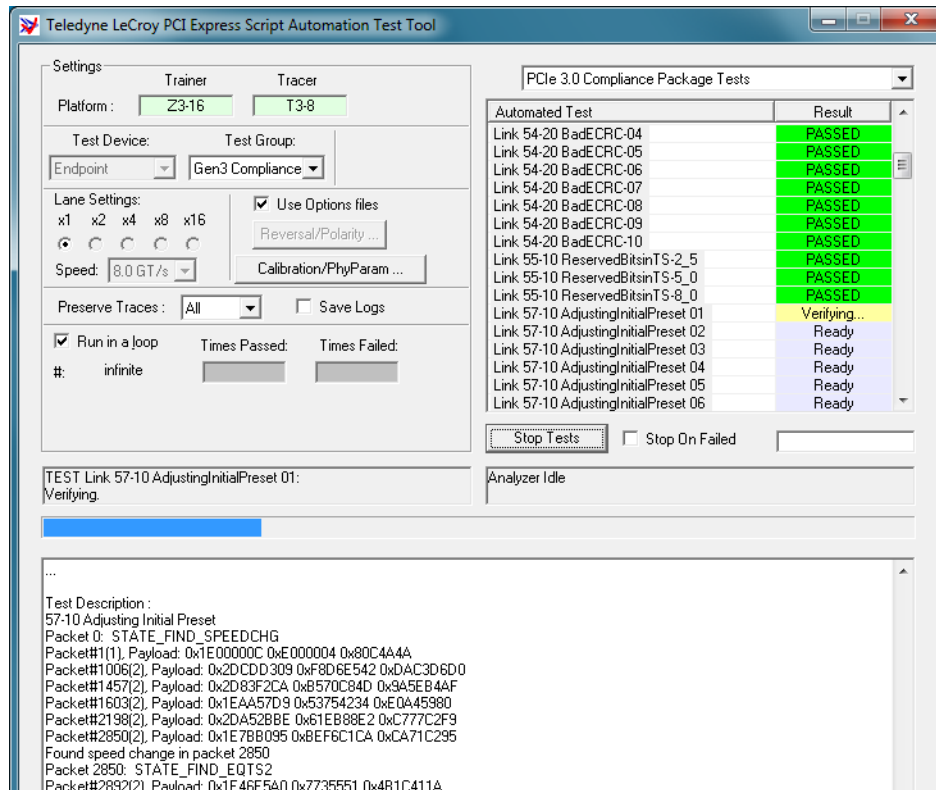


Figure 3.8: Test Results (typical).

9. Each test run will result in a new folder with the date and time of the test run.

### 3.6 Tests

The following tests are required for Official PCIe 3.0 Compliance. The PCI Express Requirements are listed in greater detail in PCI Express SIG's Test Specification.

For further information regarding the tests, refer to PCI Express SIG's "PCI Express Architecture Link Layer and Transaction Layer Test Specification, Rev. 3.0, June 6, 2013".

Table 3.1: PCI Express Tests

PCI Express Requirements		Teledyne LeCroy Tests
41-20	ReservedFieldsDLLPReceive	link_41-20_ReservedFieldsDLLPReceive
52-10	ReTransmissionNAK	link_52-10_RetransmitOnNak
52-11	ReplayTimerTest	link_52-11_REPLAY_TIMER
52-12	ReplayNumTest	link_52-12_REPLAY_NUM
52-20	LinkRetrainOnRetryFail	link_52-20_LinkRetrainOnRetryFail
52-100	ReplayTLPOrder	link_52-100_ReplayTLPOrder
52-150	CorruptedDLLP	link_52-150_CorruptedCRC_DLLP
52-160	UndefinedDLLPEncoding	link_52-160_UndefinedDLLPEncoding
52-170	WrongSeqNumberInACKDLLP	link_52-170_WrongSeqNumInAckDLLP
53-20	BadLCRC	link_53-20_BadLCRC
53-31	DuplicateTLPSeqNum	link_53-31_DuplicateTLP
54-12	TXN_BFT_RequestCompletion	trans_dut_1-2_TXN_BFT_RequestCompletion_UR: TXN_01-02
54-20	BadECRC	link 54-20 BadECRC
55-10	ReservedBitsInTrainingSequences	link 55-10 ReservedBitsInTS
57-10	Test For Adjusting Initial Preset	adjustingInitialPreset - set of 11 tests
58-10	Test For Adjusting Presets	adjustingPreset - set of 16 tests
59-10	Test For Adjusting Coefficients	adjustingCoefficients - set of 55 tests.
61-10	Check the Behavior of the DUT During FLR	FLR1
61-20	Check Whether the Physical and Data Link Layers are Reset After an FLR	FLR2-4

Table 3.1: PCI Express Tests (Continued)

PCI Express Requirements		Teledyne LeCroy Tests
62-10	Check if Upstream Port DUT Sends LTR Message After the LTR 35 Enable has been Set/Cleared and the Format of the Message	LTR1
62-20	Check if Upstream Port DUT Sends LTR Message After the Device has been Directed to Non-D0 State	LTR2-4
65-10	L1 for D3 State	link 65-10 L1ForD3State
66-10	ASPM-L1	link 66-10 ASPML1

The following test is required for official PCIe 3.0 Compliance but is run using the Teledyne LeCroy PeRT3, and is not part of this test suite:

#### **56-10 De-emphasis Request During Speed Change**

The following tests are specified but not required for official compliance testing, these tests can be run using the LTSSM test arcs from the PETracer software.

#### **60-10 Loopback Through Configuration**

- Informational Only test - not required for compliance. Use LTSSM Test arc to test.

#### **60-20 Loopback Through Recovery**

- Informational Only test - not required for compliance. Use LTSSM Test arc to test.

### **3.7 Verification of Test Compliance**

To verify that each of the tests in [Table 3.1 on page 58](#) ran at the appropriate data rate, open the trace file created by the test and the data rate will be shown. In addition, the data rate can be seen in the status bar of the Exerciser and Analyzer in the PCIe Protocol Suite software. It is recommended the the PCIe Protocol Suite software is open while running the PCI Express Script Automation test tool.





# Appendix A

## How to Contact Teledyne LeCroy

---

Type of Service	Contact
Call for technical support...	US and Canada:1 (800) 909-7112
	Worldwide: 1(408) 653-1260
Fax your questions...	Worldwide:1 (408) 727-6622
Write a letter ...	Teledyne LeCroy
	Protocol Solutions Group
	Customer Support
	3385 Scott Blvd.
	Santa Clara, CA 95054-3115
Send e-mail...	psgsupport@teledynelecroy.com
Visit the Teledyne LeCroy web site...	teledynelecroy.com/
Tell Teledyne LeCroy	Report a problem to Teledyne LeCroy Support via e-mail by selecting <b>Help&gt;Tell Teledyne LeCroy</b> from the application toolbar. This requires that an e-mail client be installed and configured on the host machine.

