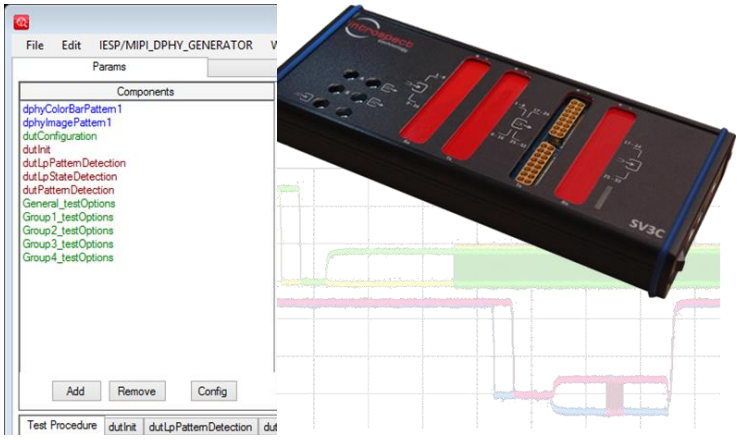
introspect technology

MIPI D-PHY 1.2 Receiver Conformance Test Suite



User Guide

Table of Contents

List of Figures

Introduction

The Introspect D-PHY 1.2 Receiver CTS Application is a Test Procedure that executes within the IntrospectESP software environment and that enables automated testing of D-PHY receivers using the Introspect SV3C DPTX 4-Lane MIPI D-PHY Generator. The Test Procedure provides a fast and easy way to validate and debug D-PHY links as well as execute automated D-PHY electrical checklists based on the MIPI Alliance Conformance Test Suite.

The Introspect D-PHY 1.2 Receiver CTS Application enhances your productivity and saves you valuable time by allowing you to focus on the device under test (DUT) specific steps of receiver testing without worrying about test system programming. It provides an easy to use interface for selecting and sequencing tests and gives you visibility into the code structure and execution flows.

The Application also includes report generation features and custom code sections that allow you to achieve true test automation – through a true software handshake. Specifically, the custom code sections are compatible with executing external scripts for controlling a device or for accessing your proprietary .NET DLL for DUT control.

Features

The Introspect D-PHY 1.2 Receiver CTS Application offers

- MIPI specification coverage for D-PHY v1.2 and beyond
- Configurable lane count and data rate
- Configurable test pattern / packet construction including arbitrary video frames
- User selection of tests based on the CTS test groups
- User-editable custom error-checking functions for true automation with DUT software
- Report generation
- Enhanced debug modes

Hardware Setup and Global Method of Implementation

Figure 1 shows the hardware setup for receiver tests in general. All lanes of Introspect SV3C DPTX generator are connected directly to the Device under Test (DUT) board. The SV3C DPTX is connected to the control Computer through a USB cable and provided software drivers; and the DUT is connected to the control Computer using its own mechanism (e.g. JTAG to USB converter). In the rest of this document, we assume that the DUT is connected to the same control Computer as the SV3C DPTX, but this is generally not a requirement for the Introspect CTS Application.

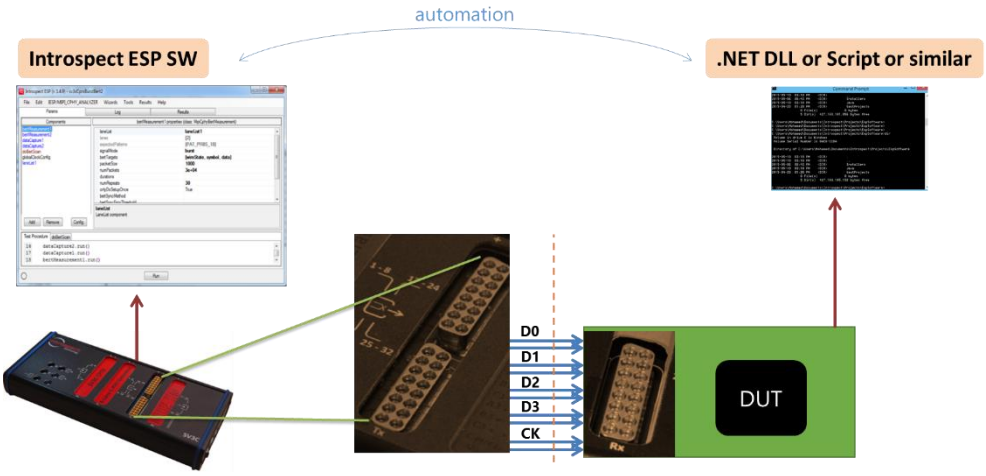


Figure 1 Overall hardware setup for D-PHY 1.2 Receiver CTS testing.

The Introspect ESP Software provides the main control for the SV3C DPTX, and the features of the CTS Application will be described extensively in later sections of this document. However, in this section, we describe the general scope of control within the CTS Application. Specifically, for each CTS test, the SV3C DPTX electrical or protocol parameters are modified repetitively and the DUT response is captured respectively for each repetition. Referring to Figure 2 which illustrates CTS Test # 2.4.3, The CTS Application uses the IntrospectESP built-in features for generating compliant video frames from the SV3C DPTX generator. It then modifies the transmitted video frames by inserting the false leader sequence at the appropriate location within a packet (defined by the CTS publication from MIPI Alliance). Finally, the CTS Application then sweeps the *position* of the false leader sequence in a loop. For each swept position, the CTS Application verifies the DUT status for pass/fail determination.

Verifying DUT status is generally device-specific, so the CTS Application exposes user-editable code windows for entering custom code. The code windows are populated by default for a semi-automatic flow in which an operator checks the device status and clicks a yes/no prompt from the IntrospectESP software. Advanced automation is described in a later section, and it generally involves one of several handshake mechanisms with the DUT control mechanism. These include:

- Introspect Test Coordinator Client/Server
- Run-time dotNET (.NET) DLL
- Command prompt script execution (e.g. .bat file)
- LabVIEW interoperability through the coordinator server or run-time DLL

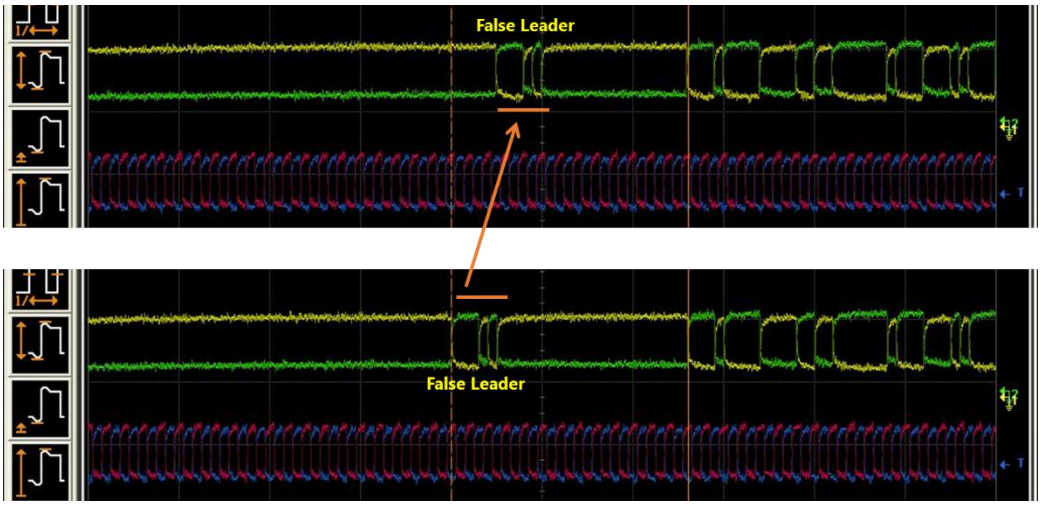


Figure 2 Overall Method of Implementation involves sweeping SV3C DPTX pattern generator parameters and checking DUT reception of image frames.

CTS Application User Interface

This section describes the components within the CTS Application. It then proceeds with instructions on how to configure tests, execute them, and customize their responses.

Component Classes Visible to the User

Figure 3 shows the Application window when it is first loaded into the IntrospectESP software. As can be seen, it is populated with a group of components that are highlighted, and these constitute the principal method for configuring tests and automating them. In the following sections, each component will be described briefly.

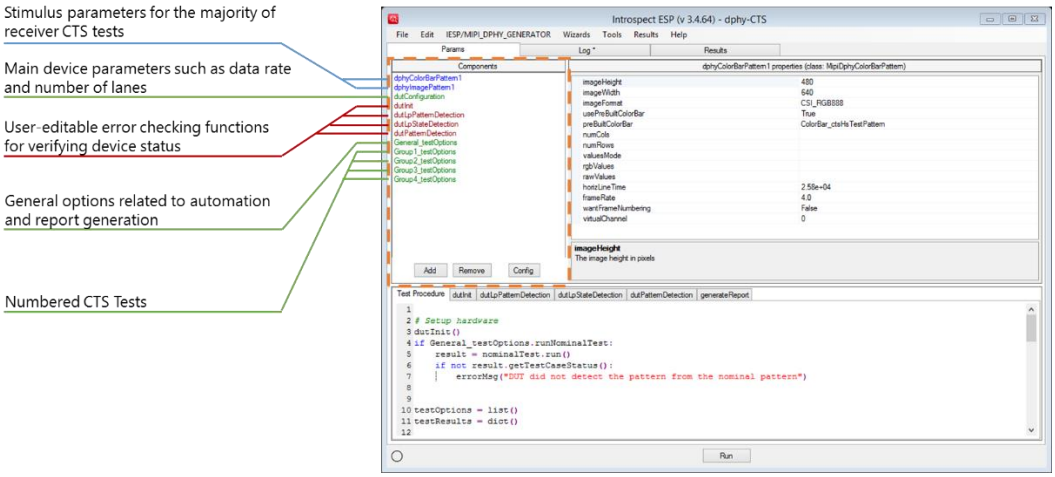


Figure 3 Illustration of the D-PHY Receiver CTS Test Procedure when opened in IntrospectESP software.

Pattern Generation Component

This component is used to define the looping video frame pattern that is required by the MIPI Alliance Conformance Test Suite document. By default, the selected pattern is the ColorBar_ctsHsTestPattern.

If a separate pattern is required, this can be instantiated as usual within the SV3C DPTX form factor. For example, an image file can be used for video frame generation, and an example of such image file is included in Figure 3 and labeled dphyImagePattern1.

DUT Configuration Component

This component is used to define main device parameters such as data rate and the number of lanes being tested. Additionally, as will be shown in a later section, this component is used to define function names for the user-editable functions that are executed when querying DUT responses automatically.

General Test Options Component

This component defines general test options related to automation and report generation. For example, it defines the behavior of the Application in case one of the tests of the CTS fails. Based on the user requirement, the entire test sequence can be aborted or the test can be skipped and subsequent tests executed. Similarly, this component is used to decide whether automated report generation is required or not.

Numbered CTS Test Selection Components

This component lists the CTS test numbers exactly the way they are published by the MIPI Alliance. Each test can be enabled or disabled based on a Boolean (True/False) variable.

User-Editable DUT Error Checking Functions

These are function tabs (visible in the Test Procedure section of the GUI) that allow you to enter custom code for verifying device response. In general, the complete CTS requires three kinds of device response checks:

- Detection of the internal LP logic state: the device is queried to report whether the LP signal on any give wire is in a high state or a low state
- Detection of a specific LP packet
- Detection of properly constructed D-PHY frame packets: properly constructed packets have many features such as proper ECC values, proper checksums, and proper payload. Depending on the device capabilities, detection of properly constructed frames can be verified in multiple ways, and this is why the function is made visible to the user

By default, the error checking functions are populated with sample code that typically includes user prompts for verifying device functionality. Thus, the CTS Application can be used immediately to debug D-PHY devices even if a software interface is not fully enabled for them.

Selecting the General Options for the Test

Figure 4 shows the General_testOptions data record component, which is used to select the options for the test. Typically, all defaults are valid. The parameter descriptions are as follows:

- runNominalTest: this commands the CTS Application to execute a quick video frame check on the device before any test is performed, thus verifying that the hardware setup is valid
- stopAfterFirstFailure: set this parameter to True if you want the CTS Application to abort after any of the tests fails
- reportGeneration: enables the creation of a CSV report of executed tests. Generated reports are accessible in the Results tab of the GUI.

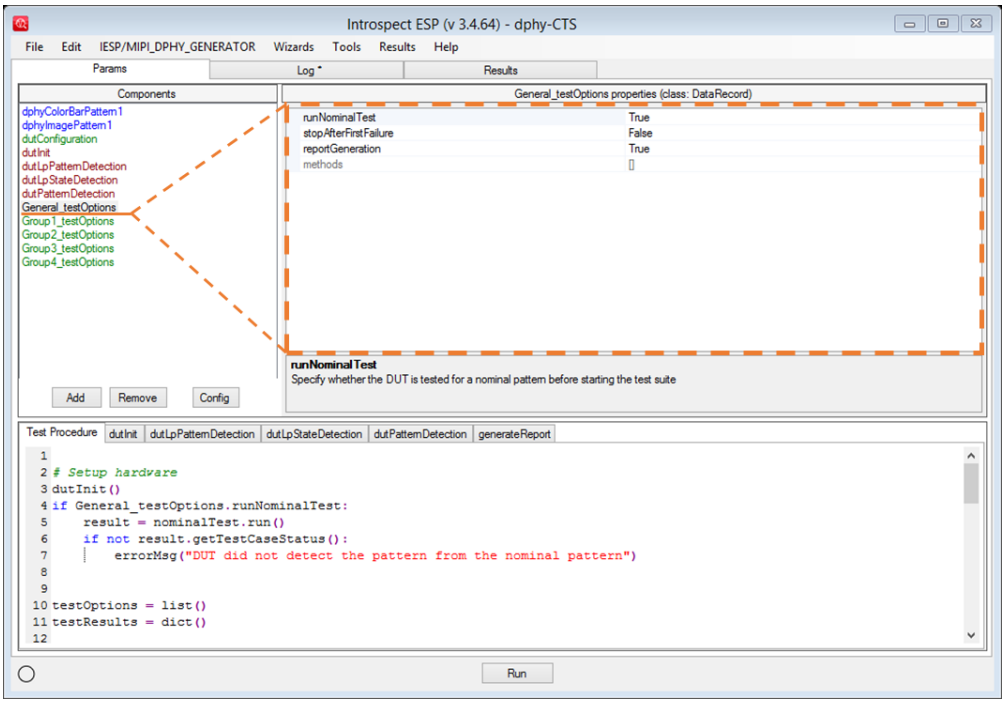


Figure 4 High-level automation options are configured using the General_testOptions data record.

Identifying Which Tests to Execute

Once the general options are established, the next step is to select which CTS tests to execute. This is done in groups as illustrated in Figure 5. In this figure, the data record for each of the groups is highlighted (by selecting it in the Components section of the GUI) in order to illustrate the list of tests for the group. Each test is enabled or disabled based on the corresponding Boolean variable.

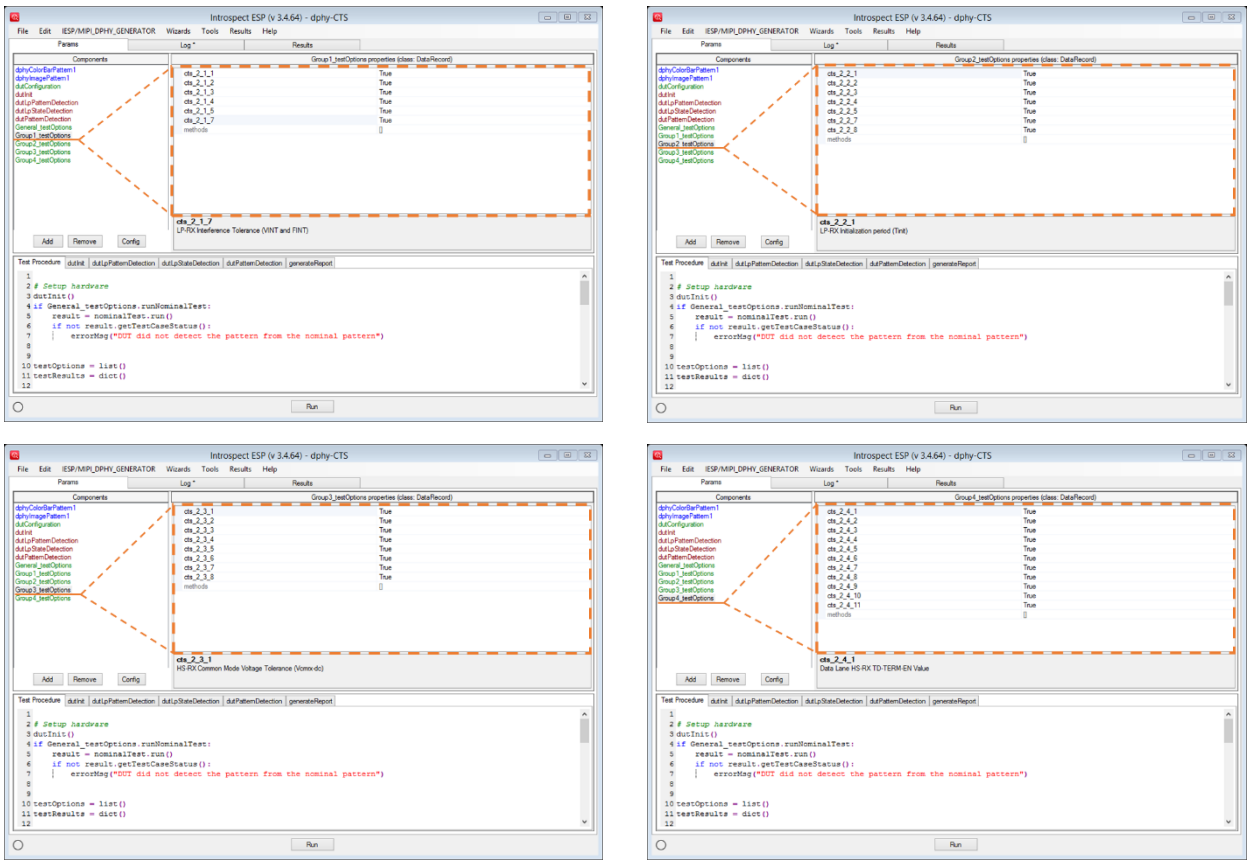


Figure 5 Illustration of the selection mechanism for each CTS test based on its corresponding group.

Defining DUT Parameters

At this point, the main execution flow for the SV3C DPTX is largely configured. What remains is to define the DUT parameters themselves, and this is done in the dutConfiguration data record component as shown in Figure 6. Select the lanes and the data rate using the top two parameters of the component. Similarly, the third parameter provides the mechanism to modify the looping frame pattern that is used for the tests in case you need to work with a custom pattern. For example, instead of selecting the default dphyColorBarPattern1 component, you can type in "dphyImagePattern1" in this parameter to select an arbitrary image payload.

The bottom half of the dutConfiguration component is used to start establishing the handshake mechanism between the IntrospectESP software and the custom DUT-specific control software. In short, these parameters constitute pointers to the custom function names described earlier.

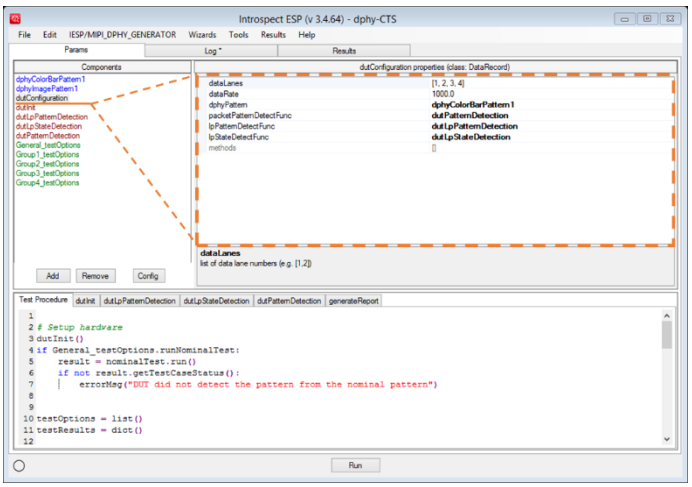


Figure 6 Illustration of the dutConfiguration component.

Defining the Pass/Fail Checking Mechanism of the DUT

Referring to Figure 7, the function names for the user-editable code sections are listed as parameters in the dutConfiguration component. This allows the user to create custom functions (using the Add button in the Components section of the GUI) without erasing the Introspect-provided sample functions.

The following sections describe these functions in more detail.

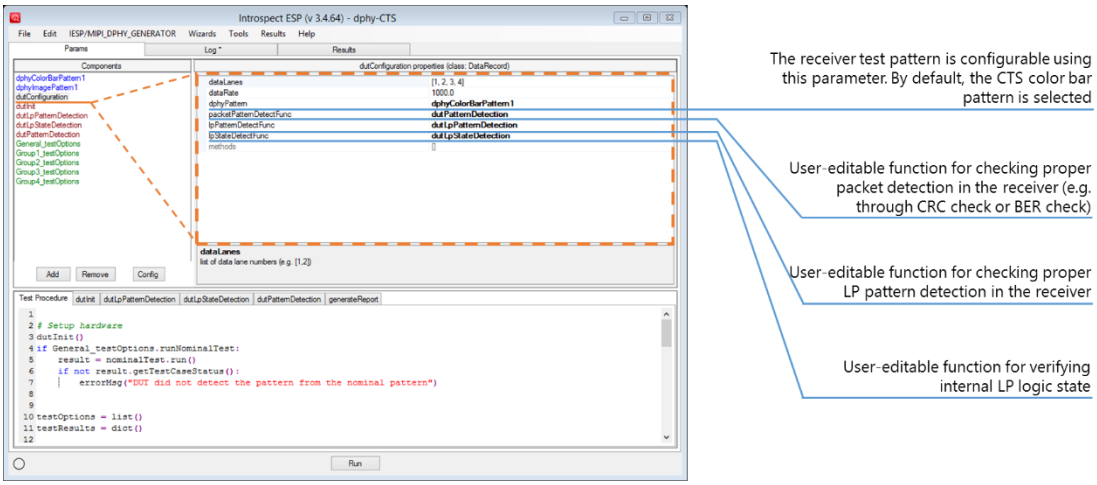


Figure 7 Defining the handshake mechanism between DUT error checkers and the SV3C DPTX D-PHY Generator.

Entering Custom Code for Initializing the DUT

Referring to Figure 8, DUT initialization is performed automatically in the CTS Application by making function calls to a function named dutInit(). This function is empty by default, but it is user-editable, allowing you to provide any code for initialization.

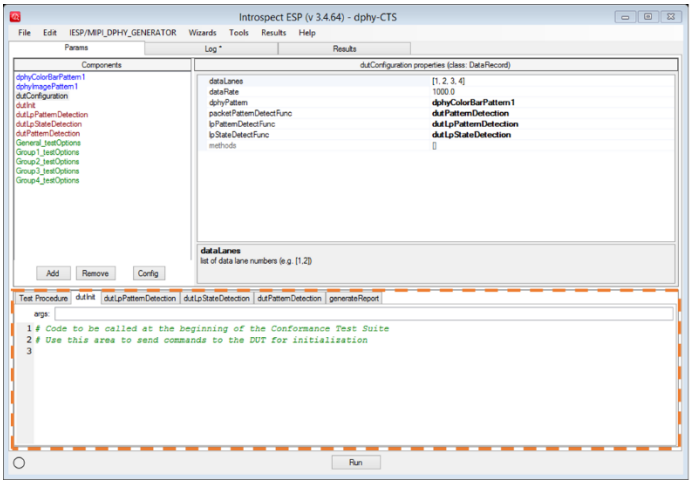


Figure 8 User-editable code area for automatically initializing the DUT or resetting it.

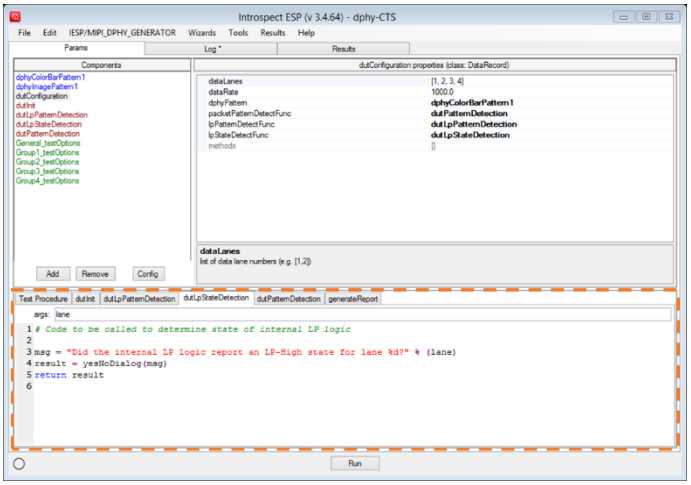
Entering Custom Internal LP State Detection Code

Apart from initialization, as seen in Figure 7, the CTS Application points the lpStateDetectFunc parameter to the function named dutLpStateDetection. This means that every time a CTS test requires the detection of the LP state on the DUT, the function named dutLpStateDetection will be automatically executed.

The dutLpStateDetection function is visible in the GUI as illustrated in Figure 9. By default, it is populated with sample code that prompts the user for a yes/no answer. The code can be modified to connect to the device and query its internal logic state automatically.

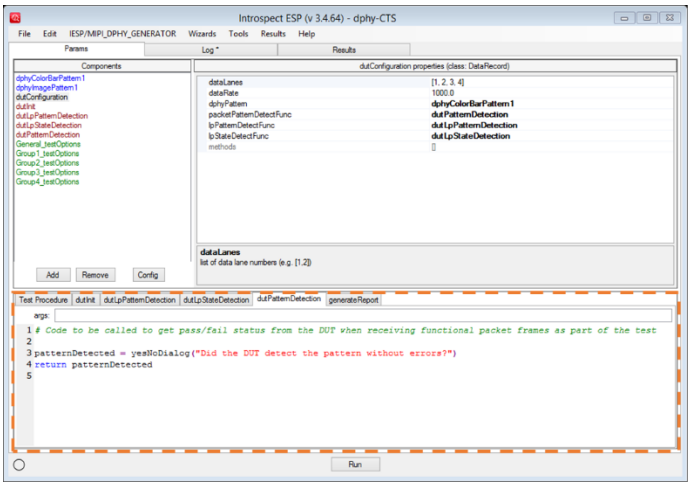
Entering Custom Functional Packet Reception Code

Similar to LP state detection, the vast majority of the CTS tests require the detection of functional packets as described earlier. Figure 10 shows the function tab for entering custom DUT checking code in the CTS Application. As with the LP state, the function is pre-populated with sample code that prompts the user for a pass/fail check on the DUT.



Sample code provided for internal LP logic state detection (uses a semi-automatic method of CTS execution)

Figure 9 User-editable function for detecting internal LP logic state inside the DUT.



Sample code provided for verifying functional packet detection (uses a semi-automatic method of CTS execution)

Figure 10 User-editable function for verifying functional packet detection by the DUT.

Automation with a Custom DUT Control DLL

In the last section, very basic pass/fail DUT checking was illustrated with the user-editable function tabs. In Figure 11, we illustrate an example of the kind of custom code that can be entered for achieving automation with a device under test. In the dutPatternDetection function, instead of prompting the user for a yes/no answer, simple Python code is entered to enable the IntrospectESP software and the CTS Application to communicate with a proprietary .NET DLL that is used to control the BER checkers within the DUT. The highlighted code sections include:

- creating an instance of the device class representing the device under test
- connecting to the device under test using the DLL-provided method
- reading the BER counter values through the DLL-provided method

It is important to note that code illustrated here is device-specific and is illustrated here in exemplary manner only. Refer to your own software automation flow (and your own DLL API) for commands that are specific to your device under test.

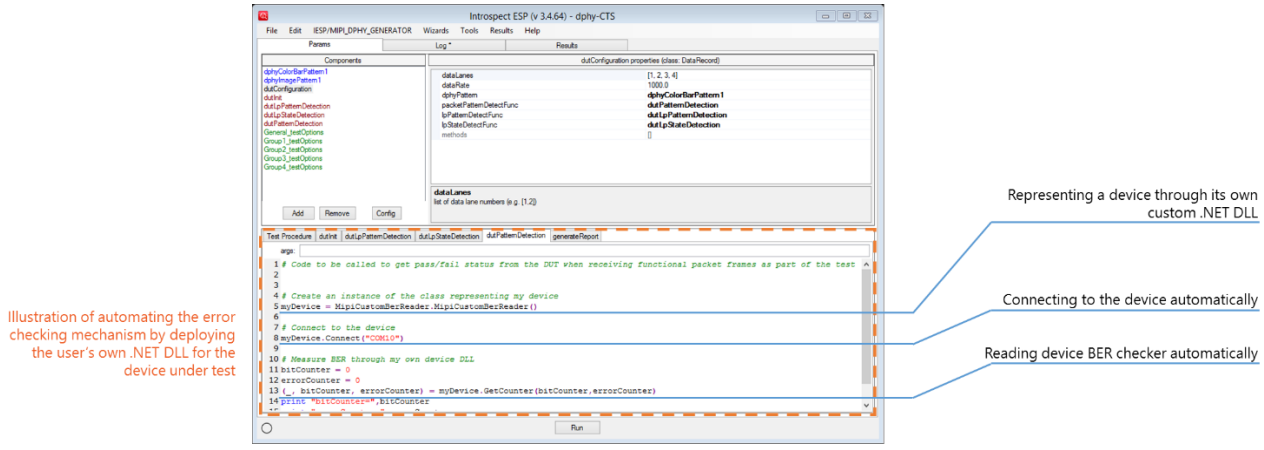


Figure 11 Custom user code (DUT-specific) for connecting to device and reading BER checker automatically.

Getting More Help

This document introduced the Introspect D-PHY 1.2 Receiver CTS Application at a high level, and it provided a detailed description of its user interface. Various other sources of information are available and these include

- Online html help files from within the IntrospectESP software
- Context help provided with each of the components and each of the parameters
- Introspect SV3C DPTX Quick Start Guide (document number EN-G005E-E-15210)
- Introspect MOI for D-PHY CTS 2.4.3 (document number EN-P001E-E-15142)
- Introspect SV3C DPTX Data Sheet (document number EN-D003E-E-15155)

Revision Number	History	Date
1.0	Document release	July 29, 2015

The information in this document is subject to change without notice and should not be construed as a commitment by Introspect Technology. While reasonable precautions have been taken, Introspect Technology assumes no responsibility for any errors that may appear in this document.

Published on July 29, 2015
EN-G007E-E-15210