introspect technology

USER MANUAL

Pinetree

Test and Development Environment

SOFTWARE

Published in Canada on April 28, 2023
EN-G040E-E-23118

INTROSPECT.CA

Table of Contents

Introduction

Pinetree is an ultra-capable development environment that allows you to easily develop and verify all your high-speed digital and mixed-signal algorithms. Designed for users with widely varying backgrounds and expertise, it offers an extremely intuitive interface simultaneously with infinitely extensible capability.

This document introduces the architecture of the software and its main features. It also provides a detailed description of its key components and presents useful programming and automation tips. Wherever appropriate, simple step-by-step procedures are presented to help you get up and running quickly with the software.

Previously known as the Introspect ESP Software, Pinetree is the software that powers Introspect's tools since the company's inception. With its most recent release comes a brand-new user interface, but with the same powerful capabilities. Pinetree was the project's original codename during the project's initial development stages. Today, it boasts its original name and reflects a pillar or strength, ingenuity, and creativity.

Requirements

Pinetree is built on the Python programming language, and it relies on four basic concepts:

- Components
- Tests
- Results
- Form Factors

The software requirements are:

- FTB2XX drivers
- FTD3 drivers (for devices with USB3)
- .NET Desktop Runtime 7.0+
- VS2017 Redistributable

- **Running as Administrator**

You must be logged in as an administrator when installing Pinetree and the FTDI drivers. If you aren't an administrator, you are likely to get install errors due to lack of permission to write files in the "system" area.

- **FTDI Driver Installation**

Pinetree communicates with the Introspect hardware/firmware via the FTDI device (connected via USB). Usually the installation of the FTDI drivers is handled automatically by Microsoft Windows when you first connect a USB cable to the Introspect hardware. If your computer is not connected to the Internet at that time, and you don't already have the FTDI "FTD2XX" and "FTD3XX" drivers installed on your computer, you will need to download them from the FTDI web site and install them using the FTDI-supplied installer.

- http://www.ftdichip.com/Drivers/D2XX.htm
- http://www.ftdichip.com/Drivers/D3XX.htm

Use the link "setup executable" on the right-hand side of the FTDI web page. Note that you must have an FTDI device plugged into your computer (via USB) for Windows to complete the install of the drivers. If you don't have access to the Introspect hardware, you will get an error message about the FTDI DLL not being installed. Ignore this if you do not intend to connect to the hardware.

After installing the FTDI drivers, we recommend using the "usbview" utility program linked to on the following FTDI page: http://www.ftdichip.com/Resources/Utilities.htm to check that your computer can "see" the FTDI device over USB.

SOFTWARE REQUIREMENTS

1) Pinetree uses Microsoft's .NET Desktop Runtime 7.0. You must have version 7.0 or later installed run Pinetree. If you don't have it installed, you can download and install the runtime manually from the link below:

- https://dotnet.microsoft.com/en-us/download/dotnet/6.0

2) Pinetree relies on the Visual C++ Redistributable (2017) library. You can download it from one of the links below:

- https://support.microsoft.com/en-ca/help/2977003/the-latest-supported-visual-c-downloads
- https://visualstudio.microsoft.com/downloads/

Pinetree requires the 64-bit (x64) version of the redistributable.

Installation

Execute the IntrospectESP_Installer.exe and follow the instructions on screen. The software can be installed on any hard drive on your computer. Note that it is not recommended to install it on a network drive as you may encounter issues with .NET 7 applications.

The installer will prompt you to select how the license should be set up on your computer. Most users will be provided with an Activation Key: select the Activation Key option to complete the installation.

You will be required to provide the activation key the first time you start Pinetree. If you do not have internet access on your computer, select the “Request a New License” option, and follow the instructions on screen.

LICENSING OPTIONS

SOFTWARE LICENSING

When you install the Pinetree GUI for the first time, in the "Software Licensing" step, you may select the option "Use Activation Key". Then, once you launch the GUI, you will be asked to enter the activation key where you can use the one provided to you.

Figure 1: Software licensing step

NOTE

Note that your PC must be connected to the Internet to enable the activation key. If you don't have access to the Internet or if you are behind a firewall, please select the option "Get a New License" instead.

To request an activation key, please create a ticket using the Introspect Help Center which is available on the support page of the Introspect Technology website: Support | Introspect Technology

If you have a previous version of the software on your PC, you may select "Use Existing License" and browse to the **Licenses folder** of your previous software installation.

You might need to get a new license with the software already installed. To obtain one, follow the steps below:

- Using the terminal or the command prompt, change the working directory to the Introspect's Python folder: `cd <your_install_path>\IntrospectESP_<version>\Python`
- Run the printInfoForLicense.py script using the command:

```
python printInfoForLicense.py
```

This script will get the hardware information of the machine you are running the software on. Please send us the information that the script generates using the Introspect Help Center.

The above steps result in the receipt of a license file that you must store in the Licenses folder of the Pinetree installation.

HARDWARE LICENSING

A hardware license is distributed as a jam file: "license_<serial number>.jam". This license is usually programmed by Introspect, and you should not have to program it unless you must upgrade your device. For example, if you originally purchased a D-PHY generator and later purchased an upgrade to C-PHY, you will receive a new jam file to enable the C-PHY license.

To program the jam file into the unit, do the following:

- Open the GUI.
- Got to Tools → License Updater
- Press start update.
- Power cycle the unit when it's finished.

The hardware license is specific to the test instrument being used and is stored inside the instrument.

For more information on licensing, refer to the Knowledgebase article: Understanding Software and Hardware Licenses

INTROSPECT DOCUMENTS FOLDER PATH

When Pinetree is installed for the first time, it automatically creates an "Introspect" folder in the Windows "Documents" directory. This folder is where Tests, Components, and Results are stored. It also contains several sub-folders for storing custom configuration data, image files, and so on. A detailed description of this Introspect folder is described in a later section of this document. For now, suffice it to say that this folder – and its contents – is always preserved when different versions of Pinetree are installed on the same PC.

Startup & Form Factor Selection

Upon starting the application, you will be greeted by a welcome screen, as seen on Figure 2. Select the Form Factor you wish you wish to use and click the *Next* button. You can either select it from the dropdown menu or type it in the Search area.

For this document, we will use the SV5C_16C12G form factor. Once selected, you will be prompted with a test selection screen: select "Create a new test" and click Next.

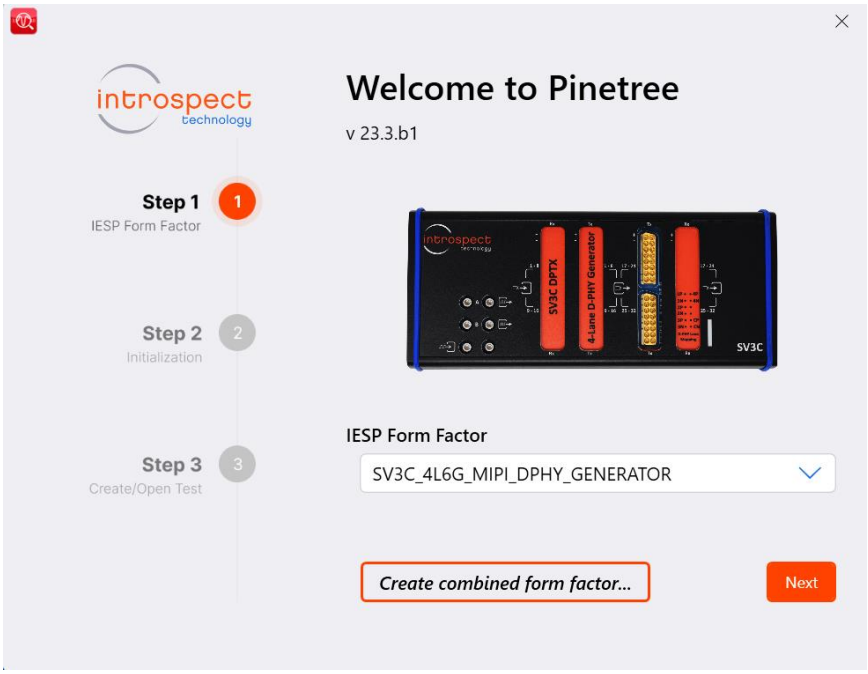


Figure 2: Pinetree welcome screen

ABOUT FORM FACTORS

A Form Factor is a software configuration that corresponds to a particular Introspect product. The correct Form Factor to choose for your product should be communicated to you by your sales representative.

Each Introspect Technology product can be driven from Pinetree by using the Form Factor associated with said product. If a user has many products, they can drive them via separate software instances. However, it is difficult to coordinate two separate software instances together for automation purposes.

Combined Form Factors allows users to drive multiple Introspect products from the same software, making it easy to coordinate two or more products together. They enable the creation of sophisticated test benches by coordinating different products to work together.

This feature is available on all platforms Pinetree is supported on (Windows, Linux, macOs).

Test Window Overview

The test window is Pinetree's main window. It shows the current software state and exposes tools the user to modify it. The major areas and buttons of the main window are listed in Figure 3.

THE GENERAL WORKFLOW OF THE SOFTWARE GOES AS FOLLOW:

- Create components as needed
- Edit the properties of each components
- Edit the test procedure
- Run the procedure. Progress or errors will be displayed in the application log.
- (optional) View the results created by the procedure

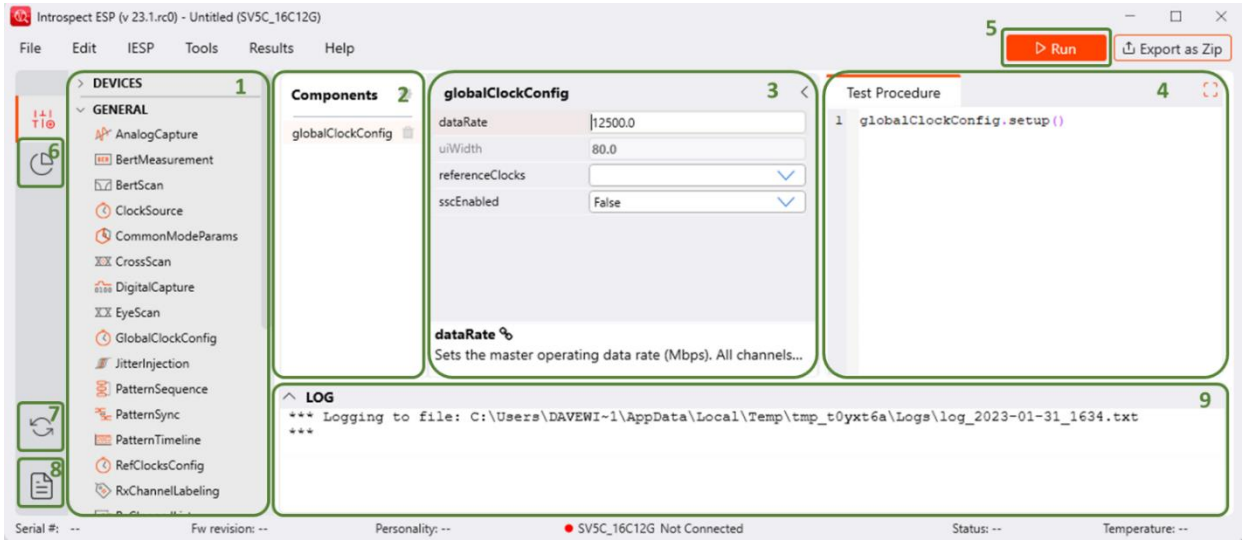


Figure 3: Main test window

1. Create new component tray (double click or drag in 2.)
2. Component list
3. Component properties panel
4. Procedure python code editor
5. Run button: start the current test
6. Result tray
7. Firmware updater
8. Link to form factor documentation (opens in web browser)
9. Application log

COMPONENTS LIST

A component encapsulates a single feature or concept of the Form Factor. Most components interact with the connected Introspect hardware in some way. Components in the "Utility" category do not directly interact with hardware but are instead tools for test organisation or automation.

The "New component tray" (area 1 of Figure 3) lists the available component classes for this test. You can create new component instances by double-clicking on the desired component. This will add a new component to the "Component list" (area 2 of figure 3).

You can have multiple instances of the same component class in your test.

Most components are used by the test procedure via either a setup() or run() method. Generally, run() methods produce a result, whereas setup() methods do not. These methods are added automatically to the test procedure upon creating a new component.

TEST PROCEDURE

The procedure is a full-fledged Python environment. From the procedure code, you can access, modify, and execute methods on the components active in the test. In addition, anything possible with a traditional Python environment is possible in the test procedure.

RUN

The run button on the top-right section executes the procedure code. At this point, you should make sure that your Introspect hardware is correctly connected to your computer via USB cable and powered on. The procedure is the sequence of events to be executed once the test has run.

The test run starts once the run button is pressed. It first connects to the Introspect hardware (if needed), then executes the procedure Python code. As the run progresses, messages will appear in the log area: if there is an error in your procedure, or an invalid setting in one of the components, a red error message will be issued, and will require a correction from the user.

Once completed, results will appear in the result tray (if applicable).

RESULTS

Certain components return a result after their execution completes. These components usually have a run() method associated with them. Most results, when double-clicked, are displayed in a viewer panel that shows all the relevant output of the component's run.

HOW TO OPEN DOCUMENTATION

You may open any documentation by following the steps below:

1. From the Components tab, right click on one of the components listed. A short list of options will appear.
2. Select Help. This will open a new web browser with the component documentation.
3. Back in Pinetree, click the chain icon next to the selected attribute's name, at the bottom. This will open component documentation for this attribute.

The main window of the application is the test window – a window that shows the Components for a Test and any Logs or Results from running that Test. Since you previously chose to create a new Test, you will be presented with an empty Test window with default Components, as shown in **Error! Reference source not found.**.

This workflow is covered in the subsections below. Additionally, an example test workflow is shown in the next section of this document.

NOTE

There is no need to press Enter after changing a property value – the change will take effect as soon as keyboard focus leaves that field. For example, you can change a property value and then immediately click the "Run" button.

Python Commands and Test Procedure Editor

The "Test Procedure" is the Python code that executes when a Test is run. Most Components have either a "run" method or a "setup" method, and a call to the appropriate method is entered automatically into the Test Procedure when you add a new Component instance to the Test.

You can add in calls to other Component methods, and comment-out or entirely remove any code that you do not want to run. You can also add in any arbitrary Python code that you want to run in the Test Procedure.

Usually, the Test Procedure is edited to reorder the calls to the Component methods so that operations happen in the right order according to your test plan. For example, if you are using the IESP's TX channels to output patterns to your device under test (DUT) and an EyeScan to analyze the Result, you need to call the TxChannelList's "setup" method before calling the "run" method of the EyeScan.

The rest of this section gives examples of more advanced features that the Python language allows you to do. You can skip it if this is the first time you are reading this manual.

COMMENTING-OUT SECTIONS OF CODE

To add a comment to the Test Procedure, you can use the Python comment character '#' – this makes Python ignore anything after that character on that line. This is often used to comment out lines of code that you do not want executed but want to keep for possible future use. It is also used to document your code for ease of readability.

It is also possible to comment-out multiple lines of code by using Python's triple quoting mechanism. If you have three quote marks next to each other, Python will ignore everything after that until the next occurrence of three quote marks in a row. You can use either the single quote character (') or the double quote character (") but you cannot mix and match.

```
globalClockConfig.setup # This is a comment
"""
This whole section is commented-out
txChannelList1.setup()
bertScan1.setup()
eyeScan1.run()
"""
txChannelList2.setup()
bertScan1.run()
eyeScan1.run()
```

1. BASIC PYTHON

PRINTING MESSAGES

You can use the Python "print" command to output messages to the Test log upon running the Test. For example:

```
print("This message will appear in the test log...")
```

To print the value of some variables along with your message, use f-strings, like so:

```
print(f"Iteration: {i} Value: {floatValue}")
```

ASSIGNING VALUES TO VARIABLES

You can assign values to variables of your choosing and then use those variables later in the Test Procedure. Many of the Component methods return a value, and it is often useful to assign this return value to a variable. For example, the "run" method of BertScan returns a Result class called BertScanResult, which contains many dictionaries and accessor functions. This Result class is not usually used in the code since its contents can be viewed via the BertScan viewer. However, during automation, you might want to do some specialized analysis on the data – something like the following function call "analyzeBertData":

```
myResult = bertScan1.run()
analyzeBertData(myResult)
```

For more information about the data returned from the Components "run" methods, please refer to the documentation available in the "Help" menu of the main software window.

2. ACCESSING COMPONENTS

You can change properties of a Component programmatically by assigning to the properties and then calling either the "setup" or "update" method on the Component. The "setup" method sends all the Component property values to the Introspect hardware, while the "update" method only sends the values that have changed. For example:

```
txChannelList1.voltageSwings = [800, 600]
txChannelList1.preEmphasis = [txPreEmphasis1]
txChannelList1.update()
```

This allows users to change Components properties on the fly during a Test run, without having to change the properties manually from the Params tab.

3. USING UTILITY FUNCTIONS

To access the TestProcedure Api, click on the documentation tab at the bottom left of the screen. A browser will open with the various procedures.

A simple example is sleepMillis: Sleep for the specified number of milliseconds.

Parameters:

- numMillisec – a non-negative integer number of milliseconds specifying the total time to sleep
- checkIntervalSec – a floating-point number of seconds specifying the interval between checks on the status of the current Test run. 'checkIntervalSec' defaults to 5 seconds. This means that by default, there will be a check on the status of the current Test run at the beginning and then every 5 seconds after that during the sleep time (if 'numMillisec' is greater than 5000).

Example:

```
>>>sleepMillis(1000)
```

4. DEFINING FUNCTIONS

Within most Introspect Form Factors, the SvtFunction Component can be added to a Test Procedure to encapsulate sections of code. Functions can be edited in their respective tab on the bottom view of the main GUI window, alongside the "Test Procedure" tab. You can define a function, "function1", as shown below in Figure.

Figure 5: Simple SvtFunction declaration

You can then call the function in your Test Procedure like so:

```
function1(1,2,3)
```

Here is a more interesting example – a simple version of what is done in the IdentifyPattern Component:

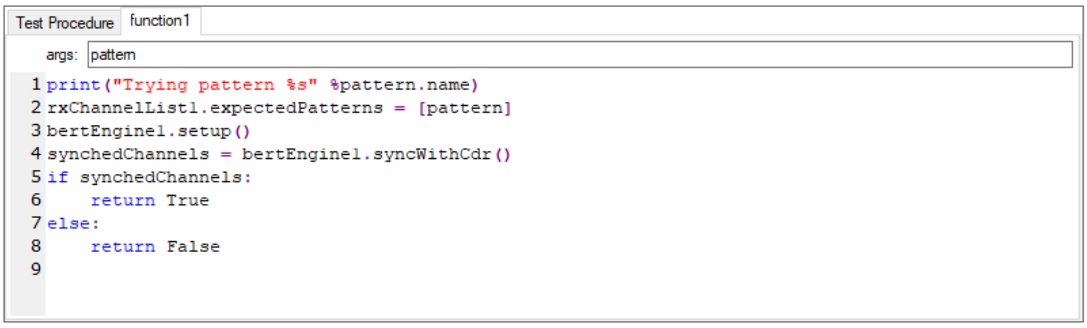


Figure 4: More advanced SvtFunction Declaration

5. LOOPING

You can loop, or iterate, over sections of code by using the Python looping constructs “for” or “while”. For example:

```
for i in range(0, 3):  # i will be 0, 1, 2
    print (f"{i}) Hello")

myPatterns = [PAT_DIV20, PAT_K28_5, userPattern1]
for i in range(0, 3):
    pattern = myPatterns[i]
    print (f"Using pattern {pattern.name}")
    rxChannelList1.expectedPatterns = [pattern]
    rxChannelList1.update()
    bertScan1.run()
```

6. USING THE STANDARD LIBRARY

Pinetree runs a fully-fledged Python interpreter. You not only have access to built-ins (such a list, tuple, etc), but also to all of python’s standard library. These can be accessed the traditional way, using import statements:

```
import os

folderPath = "path/to/folder"
fileName = "data.bin"

filePath = os.path.join(fileName)
with open(filePath, 'rb') as fh:
    dataBytes = fh.read()
```

7. USING NUMERICAL PACKAGES (NUMPY AND SCIPY)

Pinetree’s built-in interpreter comes with both Numpy and Scipy already installed. You can access them via imports:

```
import numpy as np

# Create an array of 0xAA bytes
data = np.ones(16, dtype=np.uint8)
data *= 0xAA
```

8. INTEGRATING YOUR OWN PYTHON SCRIPTS

In the above example, the implementation of the function "analyzeBertData" might have been declared in an external Python code file. Instead of copy/pasting this code into your Test Procedure, you can make it available in Pinetree in two different ways:

When a ".py" file is added to the "Params" subfolder of a Test, an instance of the PythonModule Component is automatically created with the same name as the file. By default, using the "run" method of that Component will import all functions and classes from the Python code file. This allows the user to call those functions and classes within Pinetree without having to import any extra file

By default, the Python search path of Pinetree will look for any ".py" files located in the "Documents\Introspect\PythonCode" folder. By using the Python "import" command in your Test Procedure, you can access the functions and classes declared in those files inside Pinetree. For example, if there is a "myPythonStuff.py" file in the "Documents\Introspect\PythonCode" folder, you can import all its functions and classes by using:

```
from myPythonStuff import *
```

This is especially useful when you want to use the same Python code file in multiple Tests without having to store multiple copies of the file on your computer.

INSERTING DELAYS BASED ON TIMERS

You can pause the execution of the Test Procedure for a specified number of milliseconds by using the "sleepMillis" function – for example:

```
sleepMillis(1000) # One second delay
```

INSERTING DELAYS BASED ON USER INPUT

You can pause the execution of the Test Procedure until something else is ready via the "waitForOkDialog" function – for example:

```
waitForOkDialog("Click OK when the DUT is ready")
```

This will prompt the user with a dialog box and an "OK" button. The Test Procedure will be paused until the button is clicked.

RUNNING A SHELL SCRIPT

You can run a shell script or other Windows executables via the "runShellScript" function. This could be used to control other test equipment or to do data analysis. For example:

```
runShellScript("perl myScript.pl")
```

SENDING AN EMAIL OR TEXT MESSAGE

You can send an email message at any point in the Test Procedure by using the "send" method of the EmailMessage Component. For example:

```
emailMessage1.send("Partial Result", "value: %.3f" %value)
```

NOTE

Most cellular phone companies provide a way to send text messages (SMS) to a phone via email (see for example: http://www.makeuseof.com/tag/email-to-sms/) and thus the EmailMessage component can be used to send text messages to your phone.

LOW-LEVEL ACCESS TO THE INTROSPECT HARDWARE

Usually, you control the Introspect hardware via the facilities provided by the Components themselves. But lower-level access is available via the "iesp" object that you can obtain by calling "getIespInstance()" - for example:

```
iesp = getIespInstance()
iesp.checkHardwareStatus()
```

As was mentioned previously, an IESP Form Factor is a class that provides information about the hardware configuration and that also provides methods for hardware-dependent functionality. There is usually only one instance of this class, obtained using the above call to getIespInstance().

A more sophisticated example of using low-level access is if a user wants to display the temperature of the Introspect device during their test. In this case, they could add the following lines to their Test Procedure:

```
iesp = getIespInstance()
moduleTemperatures = iesp.getModuleTemperatures()
print ("Module Temperatures: %s" %moduleTemperatures)
```

For more details on the functions mentioned see the documentation in the files “svt.html” and “iesp.html” (in the folder “Doc”). To learn more about Python, see the documentation at Welcome to Python.org

Saving and Loading Tests

Pinetree is a document-based application like Microsoft Word. Each Test window is a separate "document" and can be saved and loaded independently. You can have several Test windows open at the same time.

- When you create a new Test, the associated parameters and data are either kept in RAM or in a temporary folder on your hard disk. This allows you to do quick "one-off" experiments without being bothered about file names or folder paths.
- Just as you would on a Word document, save your test by clicking on: File and Save As, followed by naming the test to your choice. Or, press on Ctrl + S on your keyboard for a shortcut. We recommend saving your tests and associated result data for later use.
- To load a previously saved Test, click on File and Open.

STRUCTURE OF A TEST FOLDER

A Test folder always has three sub-folders: "Params", "Results" and "Logs".

- **Params** folder usually only has one file: "testProcedure.py". This file contains the Python code to create the Components of the Test with their respective properties, followed by the code of the Test Procedure. If you are careful to get the syntax correct, you can manually edit the "textProcedure.py" file in a text editor like "Notepad++" or "Komodo".
- **Results** folder is initially empty but each time you run the Test, a new sub-folder is created there to hold the results of the run. By default, these folders are named according to the date and time the Test was started. Each run Results sub-folder contains:
    - A snapshot of the "testProcedure.py" file – this file contains the state of the Test parameters used to generate the associated results data. It might be useful if you have changed the Test parameters after that run and want to revert to what it was at that time. You could revert by manually copying the "testProcedure.py" file from the run Results folder to the "Params" folder.
    - One or more Component Results sub-folders, each of which contains:
        - A file ".resultInfo.csv" with info about the Component Results. This file is normally hidden since its name starts with a dot

    - One or more CSV files with the raw data from the Component's 'run' method. Usually there is one file per channel and the channel number is part of the filename
- **Logs:** This folder contains the Test log files with the messages that appear in the Log tab. Just like the Results sub-folders, there is one log file for each session, named with the date and time of the start of the session. If you are short of disk space, you can remove old log files and any run Results that you no longer need.

Utility Components

Pinetree loads Components from the "SvtPython" folder or from the Introspect Documents Folder Path. There is a rich set of built-in Components, and these are categorized based on their usage model and functionality. The "general" category holds the general-purpose Components – these Components are often used in the implementation of other Components. Then there are other categories which are: "ui", "utility", "advanced", and "devices".

Each Component has several properties and methods. The Component properties are shown in the Properties panel of the Test window.

Most basic Components have a "setup" method and an "update" method. The "setup" method sends all the Component property values to the Introspect hardware or to any connected hardware. The "update" method sends only those property values that have changed since the last "setup" or "update", saving time when executing the Test and preserving hardware state during complex operational sequences.

Components that produce a Result typically have a "run" method. The "run" method also generally returns a value that can be assigned to a Python object. This allows the Result values to be accessed programmatically, offering more flexibility to the user. The Results can also be automatically written to a file, allowing the user to display them using the different Result Viewers of Pinetree. This means that users can execute, save, and display measurements like bathtub and eye diagrams with a single line of code.

Often, Components that perform a measurement are linked to other Components. In that case, the "run" method of the measurement Component will often call the "setup" method of the linked Components. For example, a BertScan Component's "run" method will call the "setup" method of the linked rxchannelList Component.

A list of all the available Components, along with full documentation on their respective properties and methods can be found under "Component Classes" in the "Help" menu of the main software window. You can also find documentation for a specific Component by right clicking on it in the Components List view and selecting "Help" from the contextual menu. Additional documentation can also be found in the software's installation directory under the "Doc" subfolder.

NOTE

The Components' Python class names usually start with "Svt", but this prefix is omitted in the Component class names listed in the Component List.

Below is a short list of the most common Components used in Pinetree. Before using any of these Components, it is highly recommended to quickly read the documentation about them.

UTILITY COMPONENTS

TESTCASE

Based on concepts from unit test software testing frameworks, the TestCase Component is a very powerful tool for running a sequence of operations using custom Python code that you provide to it. The Component provides utilities to define pass/fail criteria for a specific test case that you want to cover. It also offers sophisticated crash-handling capabilities, allowing you trap exceptions and to log them without halting the rest of your Test procedure.

TESTCASESUITE

A TestCaseSuite is a collection of TestCase instances. It offers methods for running test cases and for setting up global initialization parameters before running any test case. Additionally, it provides an optional graphical interface where operators can click on tests and observe pass/fail logs associated with such tests.

Please refer to the online help and tutorials for further details on the TestCaseSuite and some of its applications.

TESTASCOMPONENT

This Component provides the ability to create a user-defined Component from a Test. When you save a Test that includes an instance of this Component, a file of Python code is generated in the "TestAsComponent" sub-folder of the Test folder. This generated code defines a new Component class whose 'run' method performs the operations that are done by the Test Procedure. If you specify a DataRecord Component in the same Test entity, the fields of that DataRecord will become the properties of the new Component.

You should "hook up" the fields of the DataRecord to the properties of the Components you want them to control by adding assignment statements in the Test Procedure. After you save the Test that includes an instance of this Component, the newly generated Component will show up in the Add Component dialog in the GUI so you can create instances of this new Component in other Test windows and try out the new Component. When you have it working properly, you can copy the generated ".py" file to the "Introspect/Components" folder under the Introspect Documents Subfolder for future use by yourself or other users. The Introspect Documents Subfolder is described in a later section of this manual.

TESTASPYTHONSCRIPT

This Component provides the ability to create a standalone Python script from a Test. When you save a Test that includes an instance of this class, a Python script is generated in the "TestAsPythonScript" sub-folder of the Test folder. This script performs the operations that are done by the Test Procedure, but it can be called from any external Python environment.

TIMEIT

This Component provides a facility to measure execution times of Components or functions.

DEVICES COMPONENTS

VISAINSTRUMENT

This Component provides the basic functionality needed to control a third-party instrument that is compatible with the VISA language. It does this by deploying the pyVisa libraries from within Python. Note that all these libraries are imported automatically, so you only have to focus on the functions that you want to perform. For example, to connect to a VISA instrument, you call the method “connect()” from this Component class.

LAUTERBACHTRACE32

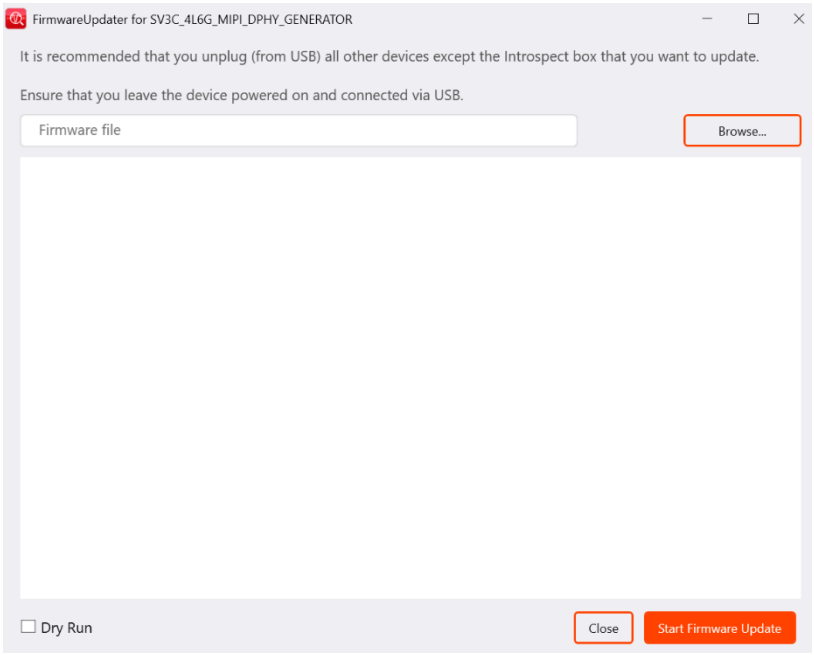
This Component provides functionality to interface with Lauterbach's Trace32 software. To communicate with a Trace32 application, it is essential that the 'packetLen' and 'port' match the values in the config.t32 used by the target application.

Updating the Firmware

To update the firmware, click on the icon above the documentation icon, in the bottom left corner of your screen (a). Alternatively, click on *Tools -> Firmware Updater* in the window top menu.

In the pop up window (b), click on "Start Firmware Update" after selecting the desired firmware file.

(a)



(b)

Figure 6: (a) Firmware icon (b) Pop up window to start the firmware update

Updating the License

To update the license, click on *Tools -> Firmware Updater* in the window top menu.

In the pop up window, click on "Start License Update" after selecting the desired license file

Customization and Preferences

When Pinetree starts up, it reads the file "preferences.ini" located in the "GUI" subfolder of the software installation directory. This file contains preferences that can be modified by the user. Here are some of the more commonly changed preferences:

`formfactor`: This specifies which Introspect hardware form factor will be shown first in the welcome screen when launching the software. For example, to display the SV1C-12 first, you would set this preference to "SV1C_8C12G" (without the quotes). The default value for this preference is MOST_RECENT.

`testDefaultPath`: This specifies the default save and load locations for all Tests. By default, the path is "Documents\Introspect\Tests". To change it, uncomment the line by removing the ';' at the start of it. Please refer to the Introspect Documents Subfolder section for further information on recommended Test paths writeMessagesToLogFile: When set to true, the messages that appear in the Log tab will be written to a file under the Test folder. This preference defaults to true.

For more details on all other preferences, please refer to the comments in "preferences.ini".

Introspect Documents Subfolder

When Pinetree is installed, it automatically creates an "Introspect" folder in the windows "Documents" directory. It contains multiple sub-folders that are shared between all installations of Pinetree, regardless of the version number. Below is a description of each of the sub-folders and their use within Pinetree.

COMPONENTS

The "Components" folder contains user defined Components. You can use the "TestAsComponent" Component in Pinetree to easily create custom Components.

CONFIG

The "Config" folder can be used to add an initialization file that applies to all installations of the Pinetree. If you modified your "preferences.ini" file under "[Introspect ESP Software Installation Directory]\GUI" as shown in the previous section, and you wish to apply it to all installations of the software on your computer, you can simply copy and paste it into the "Documents\Introspect\Config" directory.

Additionally, the Config folder is where custom form factors are stored. A custom form factor is an advanced solution for creating multi-instrument environments and controlling them from within a single instance of Pinetree. For example, you can create a custom form factor consisting of two SV5C-12 units connected together. This allows you to create very wide bus applications with a seamless software interface.

IMAGES

The "Images" folder is where users can save image files to make them accessible from within Pinetree. This allows users to easily send specific image files to their DUT when using Introspect's MIPI C-PHY or D-PHY generator devices.

LOGS

The "Logs" folder is where the generic Pinetree logs are saved. For example, this is the location where the Firmware Updater logs are saved. **This folder does not contain the log files from Test runs, it only concerns the execution of the GUI itself.**

NOTES

The "Notes" folder is where the Scrapbook entries are saved. This allows your field notes to be accessible by all the installations of Pinetree.

PYTHONCODE

The "PythonCode" folder is where you can save you external Python code files you want to use inside Pinetree. By saving a ".py" file to this folder, you can use the Python "import" command to use any of its declared function or class from within your Test. For more details on how to use your own Python code files from within Pinetree, please refer to section on **Importing Other Python Code** of this document.

SCRIPTS

The "Scripts" folder is where users can save Python scripts to be executed on raw result data. This means that you can process the raw data files produced by Pinetree to fit your exact testing needs. For more details on how to declare script files to be used inside Pinetree, please refer to the html help documentation.

TESTS

The "Tests" folder is the default location where all Tests created with Pinetree are saved. When clicking "Save" from the "File" menu in the software, this is where your Test Procedure, Logs and Results files are saved.

Troubleshooting

For detailed assistance on any issues related to Pinetree, please log-in to the Introspect Technology Service Desk and review our Knowledgebase or open a request. In this section, we provide very basic information on common troubleshooting topics.

LICENSING

Pinetree is governed by a node-locked license. New installations are delivered with an activation code that automatically generates your license files. If you do not have a network connection on the PC where the software is being installed, the activation code might not work. In this case, you will receive a license file that you have to place under the "[install path]\Licenses" folder of your installation.

If you install more than one version of Pinetree, simply copy your original license from the "[install path]\Licenses" to the corresponding folder in the new installation path.

APPLICATION STARTUP

If the software fails to start, it is likely caused by the absence of the required software libraries. See the Requirements section for the list of necessary libraries.

FAILURE TO CONNECT

Check the messages in the Log tab. The 3 most common errors encountered when failing to connect are:

- **No FTDI devices found**: the Introspect hardware is either disconnected or unpowered.
- **Command processor not responding**: Introspect hardware is hung and needs to be power-cycled.
- **A Python exception**: most likely caused by having selected the wrong form factor for the currently loaded firmware. Either restart Pinetree with the correct form factor, or update your device to the correct firmware.

Revision Number	History	Date
1.0	Document Release	January 28, 2013
2.0	Major revision and updated document template	December 9, 2019
2.1	Major revision to reflect the new GUI	April 28, 2023

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 in Canada on April 28, 2023
EN-G040E-E-23118

INTROSPECT.CA