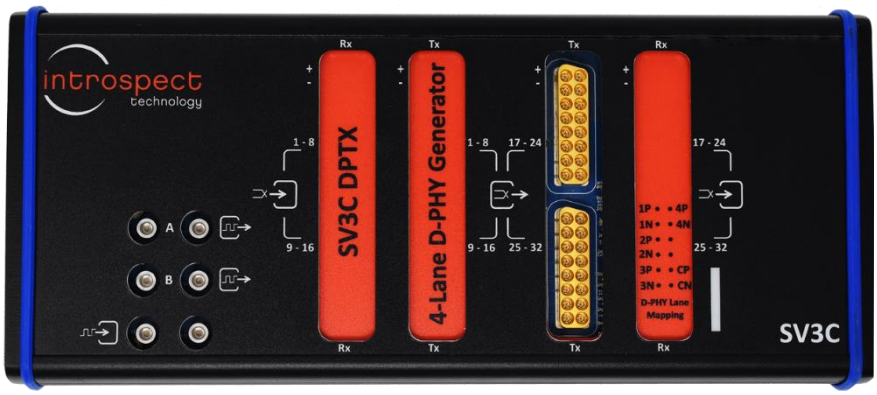
introspect technology

TECHNICAL INSIGHTS

Integrating Introspect Components with Excel VBA



Published in Canada on July 10, 2019
Technical Insights

INTROSPECT.CA

Table of Contents

Table of Contents 2
Introduction 3
Basic Concept 3
- Introspect Components 3
- xlwings Installation 3
- Using xlwings 5

Initialization 6
- Python Script 6

Illustration Example 7
- User Interface 7

Introduction

You may want to use the Introspect components within an Excel Worksheet in order to control an Introspect device. This document illustrates how you can achieve this by demonstrating how to generate a simple pattern using an SV3C-DPTX MIPI D-PHY Generator.

Basic Concept

Introspect components are written in Python; however, Excel macros do not natively support Python functions. Therefore, we propose to use the xlwings Python package. xlwings comes with an Excel add-in that allows the import of Python scripts, which can in turn be used as VBA Excel macros. For more information on xlwings and its functionality, please consult the xlwings documentation through the following link: https://docs.xlwings.org/en/stable/index.html#

INTROSPECT COMPONENTS

To be able to interact with the Introspect hardware using Excel, the Introspect ESP Software should be installed on the machine. The needed components are under the IntrospectESP_<version>\SvtPython folder of your installation.

XLWINGS INSTALLATION

Note: If you have previously installed xlwings on your machine, this procedure will override it.

In order to use Python scripts as Excel VBA macros, we first need to add the xlwings Python package to the Introspect ESP Python environment. To do so, download the "introspect_xlwings.7z" from Introspect and extract its "wheel" folder to your IntrospectESP\PythonEnv installation folder (e.g. C:\Introspect\IntrospectESP_<version>\PythonEnv). Then, open a Windows command prompt terminal and enter the following commands:

```
cd C:\[Your IntrospectESP Installation Folder]\PythonEnv
python.exe -m pip install wheel
python.exe -m pip install .\wheel\comtypes-1.1.7-py3-none-any.whl
python.exe -m pip install xlwings
```

Now that we have added the xlwings package to the Introspect ESP Python Environment, we need to install the xlwings Excel add-in.

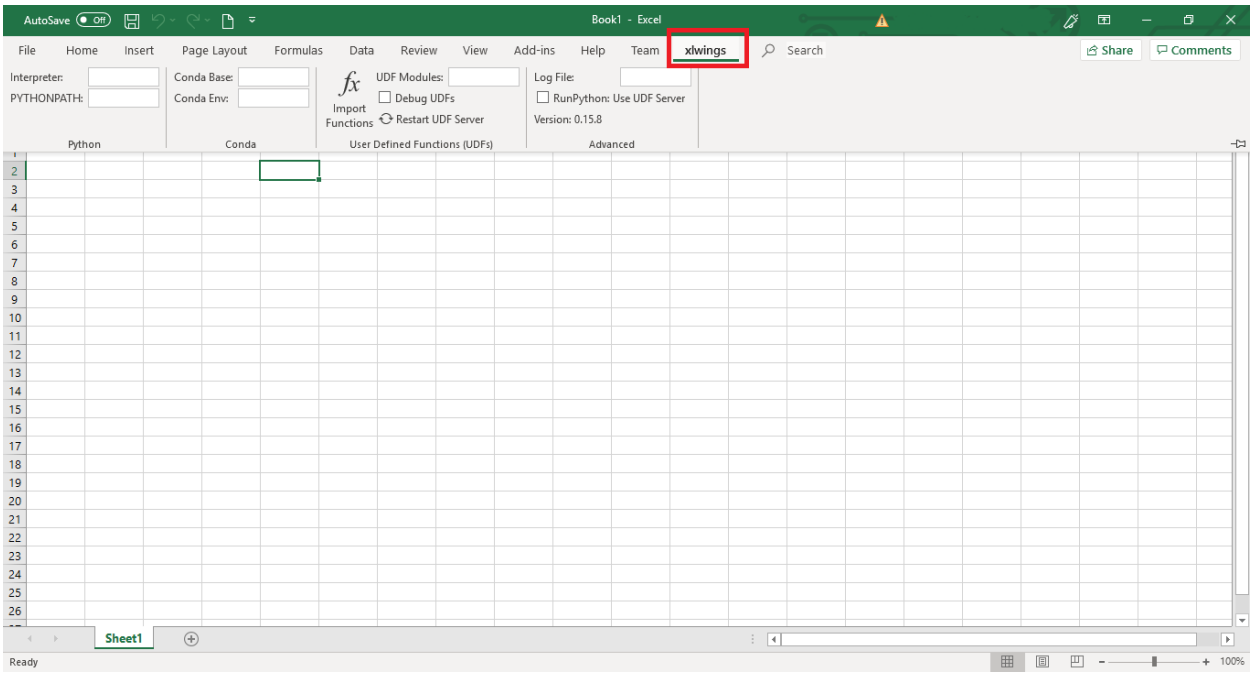
If you have previously installed xlwings on your computer, you will need to run the following command. Otherwise, you can skip this step:

```
xlwings addin remove
```

To install the xlwings Excel add-in, make sure Excel is closed and enter the following command in the previously opened Windows command prompt:

```
.\Scripts\xlwings.exe addin install
```

You can now close the Windows command prompt terminal. Open Excel, create a new Worksheet, and make sure the xlwings ribbon has been added to Excel, as below:



In order to use the Introspect ESP Python environment, we need to point xlwings to it. To do so, click the xlwings ribbon in Excel and add the following to the "Interpreter" field:

```
C:\[Your IntrospectESP Installation Folder]\PythonEnv\python.exe
```

We also need to point xlwings to the Python components used by the Introspect ESP Software. For this, we need to add the following to the "PYTHONPATH" field:

```
C:\[Your IntrospectESP Installation Folder]\SvtPython
```

Finally, we need to allow xlwings to access to the VBA project object model. Open Excel, then navigate to File>Options>Trust Center>Macro Settings and tick the "Trust access to the VBA project object model" box.

USING XLWINGS

To better illustrate the use of xlwings, we first present a basic "Hello World" example. This example uses a macro enabled Excel Worksheet "hello.xlsm" along with an external Python script "hello.py":

hello.py

```
import xlwings as xw

@xw.func
def hello()
    wb = xw.Book.caller() #Gets the name of the WorkBook
    wb.sheets[0].range('A1').value = "Hello!" #Prints Hello! to cell A1
```

NOTE

Both files have to be located in the same directory and have the same name in order to use the "Import functions" functionality of xlwings. If both files do not have the same name, or if you need multiple Python scripts to be supported by xlwings, add their name to the "UDF Modules" field of the xlwings Excel ribbon.

To import the "hello()" function into Excel, open "hello.xlsm", click the xlwings ribbon and click the "Import functions" button. Now, we need to add xlwings to the VBA Reference. To do so, open the Developer Console (Alt-F11), click on Tools>References and select xlwings from the list.

You can test the macro by typing adding a macro button using the "hello" macro we just added. When the button is clicked, cell A1 will change its value to "Hello!"

Initialization

PYTHON SCRIPT

In order for the Introspect hardware to communicate with the Introspect ESP Software, you need to initialize the necessary components for your script. Below is an example of how prepare for the generation of a pattern on the Introspect SV3C-DPTX MIPI D-PHY Generator:

introspect.py

```
import xlwings as xw
import sys, os
from dftm.svt import initFormFactor, createComponentContext

formFactorName = 'SV3C_4L6G_MIPI_DPHY_GENERATOR' #To be changed depending on which form factor
you use
iesp = initFormFactor(formFactorName)
svtContext = createComponentContext()
svtNamesDict = svtContext.getNamesDict()

globalClockConfig = svtContext.createComponent("SvtMipiClockConfig")
dphyCsiColorBarPattern1 = svtContext.createComponent("SvtMipiDphyCsiColorBarPattern")
dphyParameters1 = svtContext.createComponent("SvtMipiDphyParameters")
mipiDphyGenerator1 = svtContext.createComponent("SvtMipiDphyGenerator")
mipiDphyGenerator1.dphyPattern = dphyCsiColorBarPattern1
mipiDphyGenerator1.dphyParams = dphyParameters1


def get_workbook_name():
    wb = xw.Book.caller()
    return wb

@xw.sub
def connectToIESP():
    connected = iesp.connectViaFtdi()

@xw.sub
def disconnectFromIESP():
    # disconnect
    iesp.disconnect()

@xw.sub
def setupTxChannel():
    mipiDphyGenerator1.setup()

@xw.sub
def startCtsTest():
    wb = get_workbook_name()
    lpVoltageHigh = wb.sheets[0].range('B16').value
    lpDataHighVoltages = [lpVoltageHigh]
    setupTxChannel()
```

Before creating the components, an initialization is needed (initializing the form factor and creating the context for the Introspect components); please refer to the Using Introspect Components in IronPython Scripts article for further explanation.

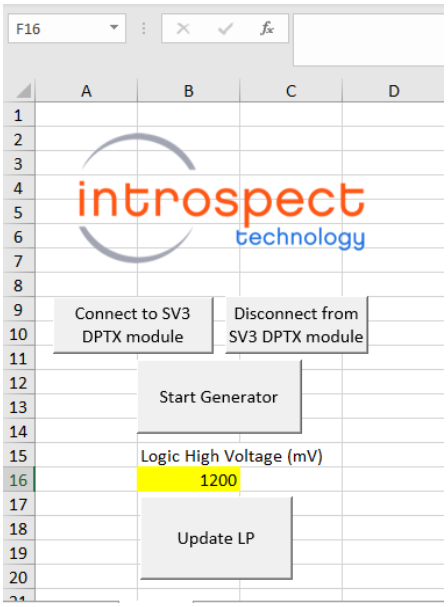
In the "introspect.py" file, we define the Introspect components that will be used for our test. Note that this script is an illustration. You may need to modify it or create your own script depending on your testing requirements. All the available components are defined and documented under the Introspect<version>\Doc\<FormFactor>\svt.html. You can also access this file by choosing the "Components classes" under the Help menu in the Introspect ESP GUI.

Illustration Example

We illustrate how Introspect components and testing procedures can be called within Excel by generating a pattern at different LP levels.

USER INTERFACE

The main user interface looks like below:



RUNNING THE GENERATOR

The xlwings Python package allows for the easy integration of Python scripts to VBA macros. By including the required software components and by defining functions in the "introspect.py" script, the user can call some pre-built Introspect Python functions to control the Introspect ESP instrument. For example, if the user wants to setup the generator, he can define a function that calls the mipiDphyGenerator1.setup() method. Note that "mipiDphyGenerator1" is the object name we set in the "introspect.py" script when creating a SvtMipiDphyGenerator component. Similarly, one can change the default parameters of the generator and then do mipiDphyGenerator1.update(). In this example, we change the LP high voltage for all data lanes to 800 mV. At the end, we perform a "reset" to the generator. Once the functions have been defined in the "introspect.py" Python script, the user can, for example, add a button to the Excel Worksheet to call those functions in a very straightforward way.