Power Supply Test Sequence

1.1 Purpose

Power Supply Tests source the Voltage from Power (TS-15200) Resource and measures all Test points in the PCB simultaneously with Analog Input (TS-15100) Resource. This example sequence can be executed in a custom Python sequence script using the measurement libraries written in Python.

This sequence uses simulated TS-15200

Example File Location

"\<venv>\Lib\site-packages\nipcbatt\pcbatt_automation\power_supply_tests"

1.2 Highlighted Features

- Power Supply Test (with Trigger)
  - Sources the supply voltage using Power Resource and measures the Test Points simultaneously with Analog input resource. Hardware Triggers are used to reduce the delay between Source and Measure. Libraries used in the example are "***PowerSupplySourceAndMeasure()***" and "***TimeDomainMeasurement()***"
- Power Supply Test (without Trigger)
  - Sources the supply voltage using Power Resource and measures the Test Points sequentially with Analog input resource without using triggers. Libraries used in the example are "***PowerSupplySourceAndMeasure()***" and "***DcRmsVoltageMeasurement()***"
- Power down all Power Supplies
  - Powers down all Power Supplies by disabling the output.

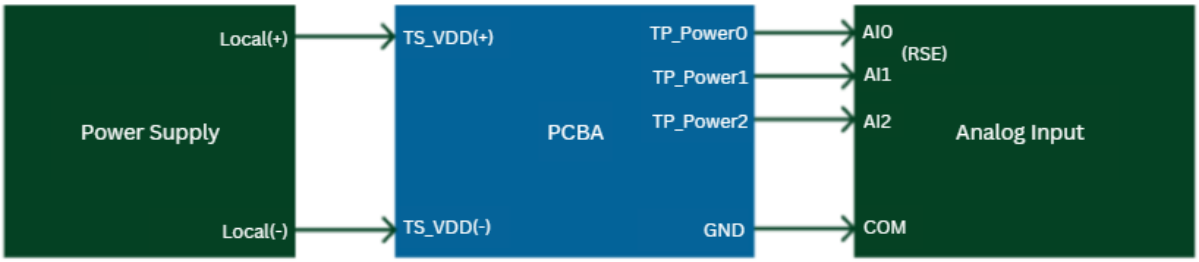
Refer this folder for more details on each Measurement library "\<venv>\Lib\site-packages\nipcbatt\pcbatt_library".

1.3 Prerequisites

- Python – 3.9 to 3.12
- DAQmx Driver – 2023 Q3 or later

1.4 Setup Diagram

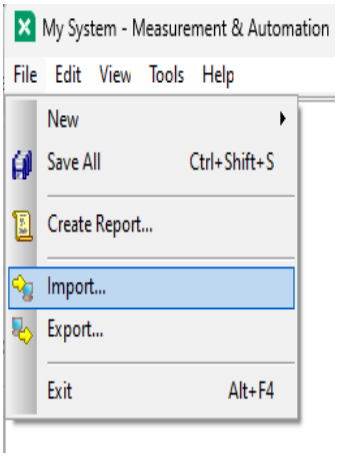
Represents the hardware used in this example sequence.



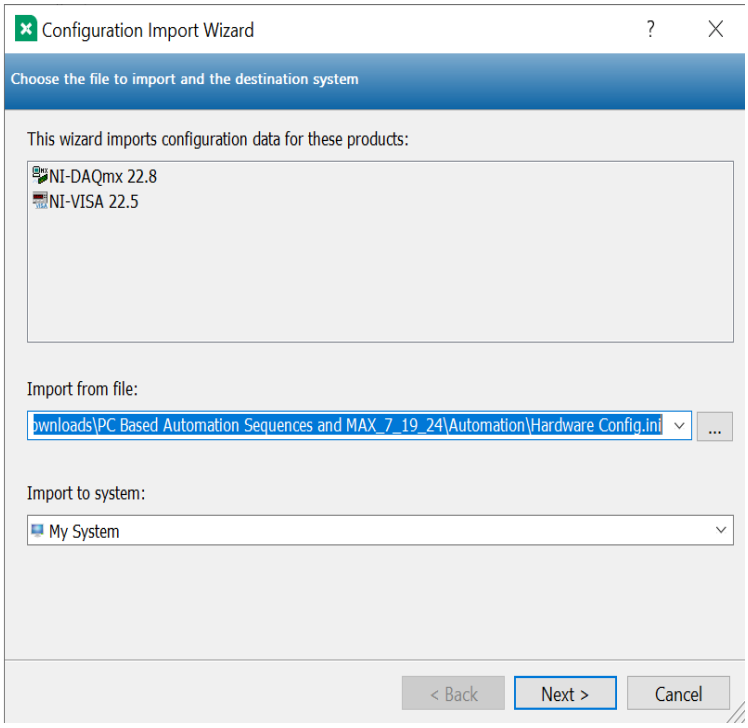
1.5 How to run this Example?

Complete the following steps to run the sequence.

1. First, we must configure NI-MAX to reflect the simulated virtual channels which will be used by the Python script names mentioned in ***power_supply_test_with_trigger.py or power_supply_test_without_trigger.py :***
    a. A hardware configuration file for NI-MAX is required to run this example. The configuration file contains a set of predefined global channel names which are used by the nidaqmx driver to communicate with the Python scripts.
    b. To import the "Hardware Config" open NI-MAX.
    c. Click on File -> Import to open the Configuration Import Wizard



d. In the Configuration Import Wizard window, click on the Browse (...) button and locate the *Hardware Config.ini* file in *"\<venv>\Lib\site-packages\nipcbatt\pcbatt_automation".* Then click on *Next -> Import -> Finish*



e. NI-MAX now holds the same virtual channel name references contained in the examples provided
    a. The ***analog_voltage_measurement_test.py*** and ***analog_pwm_test.py*** files will create log files in the form of a simple text (.txt) file. The default file path it will use is

The ***power_supply_test_with_trigger.py and power_supply_test_without_trigger.py fil***es will create log files in the form of a simple text (.txt) file. The default file path it will use is

"C:\\Windows\\Temp\\power_supply_test_with_trigger_results.txt"
"C:\\Windows\\Temp\\power_supply_test_without_trigger_results.txt"

If you wish to create this file in a different location on your PC, change the value of the string variable *DEFAULT_FILEPATH*.

2. Open the Python scripts ***power_supply_main_sequence.py*** *along with* ***power_supply_test_with_trigger.py*** and ***power_supply_test_without_trigger.py*** in your IDE or text editor of choice. The following steps are performed within ***power_supply_main_sequence.py.***

a. **Power Supply test (with Trigger)** - Hardware Triggers are used to reduce the delay between Source and Measure, hence capable of measuring the ramp voltage in Test Points immediately after source. Below are the steps included in the test.
    i. Initialize Power Supply and Time Domain Measurement Libraries by creating instances of ***PowerSupplySourceAndMeasure()*** and ***TimeDomainMeasurement()*** classes and then using ***initialize()*** methods on each object.
    ii. Configure Time Domain Measurement to wait for Start Trigger from Power Supply Resource.
    iii. Configure the Power Supply Source and start sourcing. Here the example **supplies 5V**. (in the backend, Power Supply resource sends the Trigger in the backplane once the Source started which in turn starts the measurement in Analog Input resource)
    iv. Fetch the Voltage measured and returns the measurement.
    v. Use the ***close()*** methods on both instances to close all tasks and release resources allocation.

    Refer the help/comments in the sequence for more details to know more about trigger configuration.

b. **Power Supply test (without Trigger)** – Demonstrates the Power Supply Source and DC-RMS Voltage Measurement without triggers. Below are the steps included in the test.
    i. Initialize Power Supply and DC-RMS Voltage Measurement Libraries by creating instances of ***PowerSupplySourceAndMeasure()*** and ***DcRmsVoltageMeasurement()*** classes and then using ***initialize()*** methods on each object.
    ii. Configure the Power Supply Source and start sourcing. Supply Voltage will be sourced in the output even after closing the task. Here the example **supplies 5V**.
    iii. Measure the sourced DC Voltage with ***DcRmsVoltageMeasurement()*** class instance by calling the ***configure_and_measure()*** method.

iv. Use the ***close()*** methods on both instances to close all tasks and release resources allocation.

c. **Power down all Power Supplies** – Powers down all Power Supplies by disabling the output. Below are the steps included.

i Initialize an instance of ***PowerSupplySourceAndMeasure()*** class to call the ***initialize()*** method.

ii Configure the power supply to source 0 Volts and Output status to "Disable Output when Task stopped" in specified Power channels by calling the ***configure_and_generate()*** method using the default parameters *DEFAULT_POWER_SUPPLY_SOURCE_AND_MEASURE_CONFIGURATION*.

iii Finally use the ***close()*** method to close all tasks and release resources allocation.

3. When the execution completes, **review the results** on the ***.txt*** files generated by the logger at the specified location.

a. The report has the configurations and Measurement values captured (runs with simulated instrument by default)

b. Verify the Measurement and data formats returned by the Measurement library

1.5.1 How to enable the Hardware?

Power Supply test sequence runs with simulated hardware by default. Once the hardware setup is available, you can do the below changes to enable running the test with the hardware.

Note: In this example, physical and global virtual channels are used to configure the terminal or pin to perform the instrument actions. Global Virtual Channels are software entities that encapsulate the physical channel along with other channel specific information—range, terminal configuration, and custom scaling. Global Channels can be created in NI-MAX and called in Measurement Libraries.

1. Skip the first step which imports simulated virtual channels in MAX as in section 1.5. If already done, you can simply update the channel names (physical or virtual) in the ***initialize()*** step of each automation sequence to match the hardware connected/detected in NI-MAX.
   Note: Please ensure correct trigger sources as mentioned in the steps below.

2. Follow the below steps for each sequence. Refer "**Note to run with Hardware**" labels in the sequence.

   i. Power Supply and Measure (with Trigger) Sequence

   1. Step into the "***power_supply_test_with_trigger.py***" sequence
   2. Update the *"POWER_CHANNEL"* input with Power Resource Channel in the ***initialize()*** method of ***PowerSupplySourceAndMeasure().***
   3. Open NI-MAX and update the physical Channel linked to the Global Channels **- TP_Power0, TP_Power1, TP_Power2** (called in the ***initialize()*** method of ***TimeDomainMeasurement()***)
   4. Update the "*digital_start_trigger_source*" based on NI-MAX Hardware in the *"DigitalStartTriggerParameters*" configured in the ***configure_and_measure()*** method
   5. Ensure the *"sample_timing_engine"* configured in *"SampleClockTimingParameters"* Matches with Trigger line defined in previous step
   6. Set the *"MeasurementAnalysisRequirement"* input to *"PROCEED_TO_ANALYSIS"* in *"MeasurementOptions"* property to perform time domain analysis in the measured data
   7. Review the Configurations of Power and Analog Input Pins for the intended use case

   ii. Power Supply and Measure (without Trigger) Sequence

   1. Step into the "***power_supply_test_without_trigger.py***" sequence
   2. Update the *"POWER_CHANNEL"* input with Power Resource Channel in the ***initialize()*** step of ***PowerSupplySourceAndMeasure()***
   3. Open NI-MAX and update the physical Channel linked to the Global Channels **- TP_Power0, TP_Power1, TP_Power2** (called in the ***initialize()*** method of ***DcRmsVoltageMeasurement()***).
   4. Review the Configurations of Power and Analog Input Pins for the intended use case

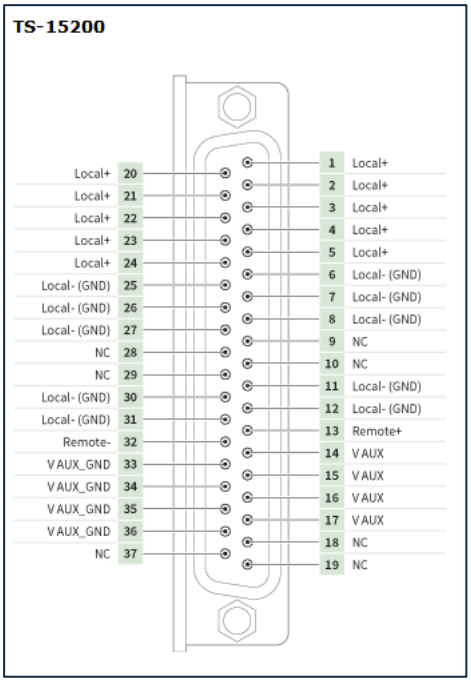
iii**. Power down all Power Supplies Sequence**

1. Step into the "***power_down_all_power_channels.py***" sequence
2. Update the *"POWER_CHANNEL"* input with Power Resource Channel in the ***initialize()*** step of ***PowerSupplySourceAndMeasure()***
3. Review the Configurations of Power for the intended use case

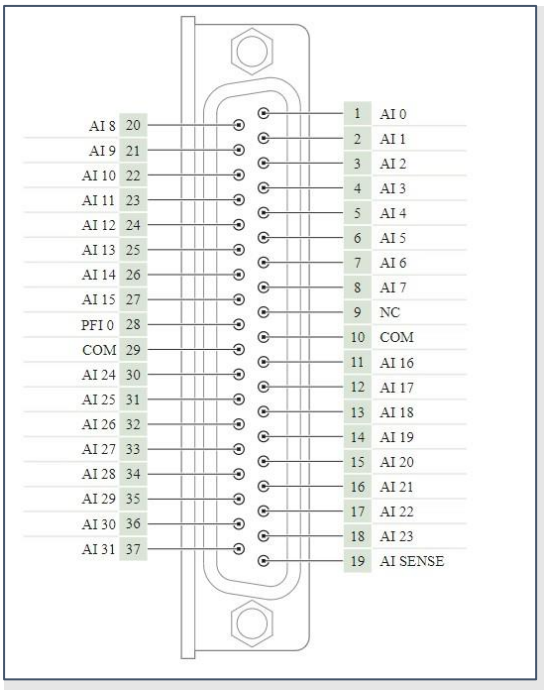
3. Review the **Source and range settings of Power and Analog input Pins** based on the DUT and Connections before running with Hardware

1.6 Pinout of TestScale Modules

1. **Power Module (TS-15200)**



2. **Analog Input Module (TS-15100)**



1.7 How to create/Modify Global Virtual Channels?

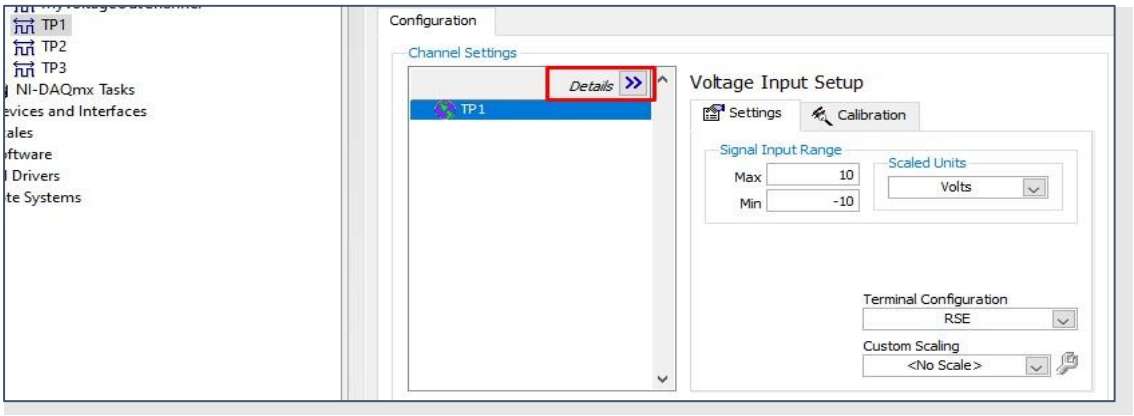
A virtual channel is a collection of settings such as a name, a physical channel, input terminal connections, the type of measurement or generation, and can include scaling information. A virtual channel created outside a task is a Global Virtual Channel.

Follow the below steps to **create Global Virtual Channel** in NI-MAX.

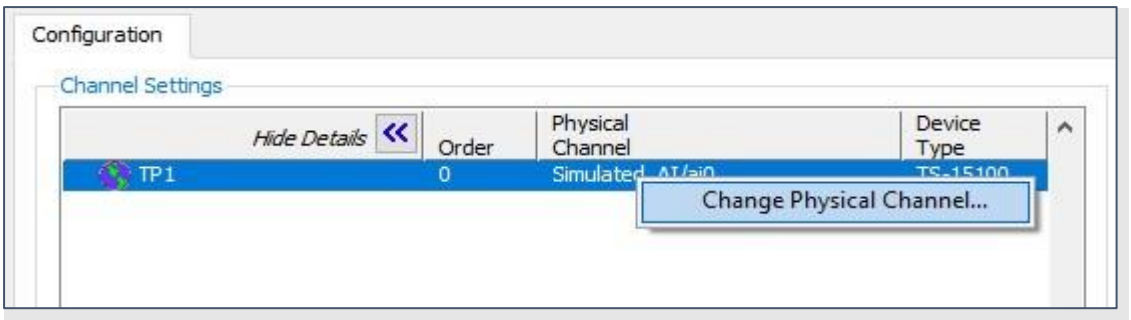
1. Launch NI-MAX
2. In NI-MAX, right-click **Data Neighbourhood** and select **Create New**
3. In the Create New window, select **NI-DAQmx Global Virtual Channel** and click **Next**. The DAQ Assistant opens.
4. Select an I/O type, such as analog input
5. Select the physical channel of Hardware
6. Type the global virtual channel name. Click **Finish**
7. Save your configuration.

Follow the below steps to **modify the existing Global Virtual Channel** in NI-MAX.

1. Launch NI-MAX
2. In NI-MAX, expand **Data Neighbourhood** > **NI-DAQmx Global Virtual Channel**
3. Select the Global Channel to modify. Configuration window opens.



4. Click on “Details >>” as highlighted above to view the Physical Channel
5. Right click and **Change Physical Channel** to update the Physical Channel. Select the Physical Channel from Hardware as per the connection and Click “Ok”



6. **Save** your configuration