LED Test Sequence

1.1 Purpose

LED Test demonstrate the measurement of differential voltage across Anode and Cathode of LED and the validation of PWM signal with Analog Buffer using Analog Input and Analog Output Resources. This example sequence can be executed in Python using the Measurement libraries in PCBA.

Example File Location

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

1.2 Highlighted Features

- Analog Voltage Measurement Test
  - Measures and validates the differential Voltage across Anode and Cathode of LED with Analog input resource. Library used in the example is "***DcRmsVoltageMeasurement()***"
- Analog PWM Test (Analog Buffer)
  - PWM signals are sent to LED circuit with Analog Output resource and the PWM response is measured in the TestPoint simultaneously with Analog Input resource using Trigger. Libraries used in the example are "***SignalVoltageGeneration()***" and "***TimeDomainMeasurement()***".
- Turn Off all AO Channels
  - Powers down all analog output channels by configuring the output voltage as 0 Volts. Libraries used in the example is "***DcVoltageGeneration()***".

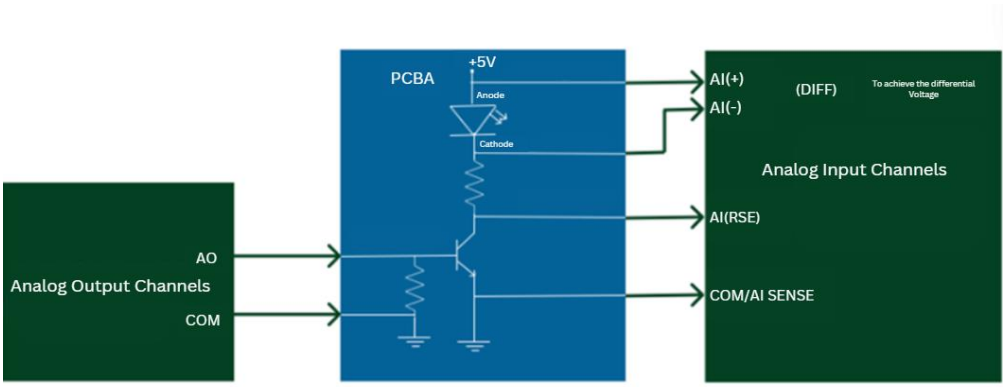
Refer to this folder for more details on each Measurement library *"\<venv>\Lib\site-packages\nipcbatt\pcbatt_library".*

1.3 Prerequisites

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

1.4 Setup Diagram

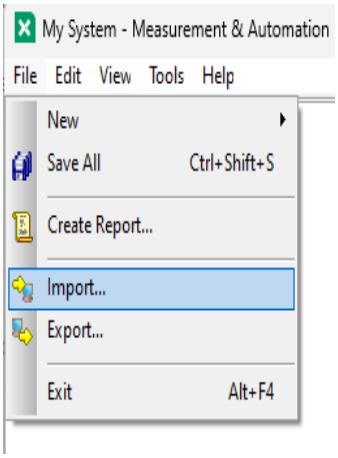
Represents the hardware used in this example sequence. Pin Outs of each resource is added below.



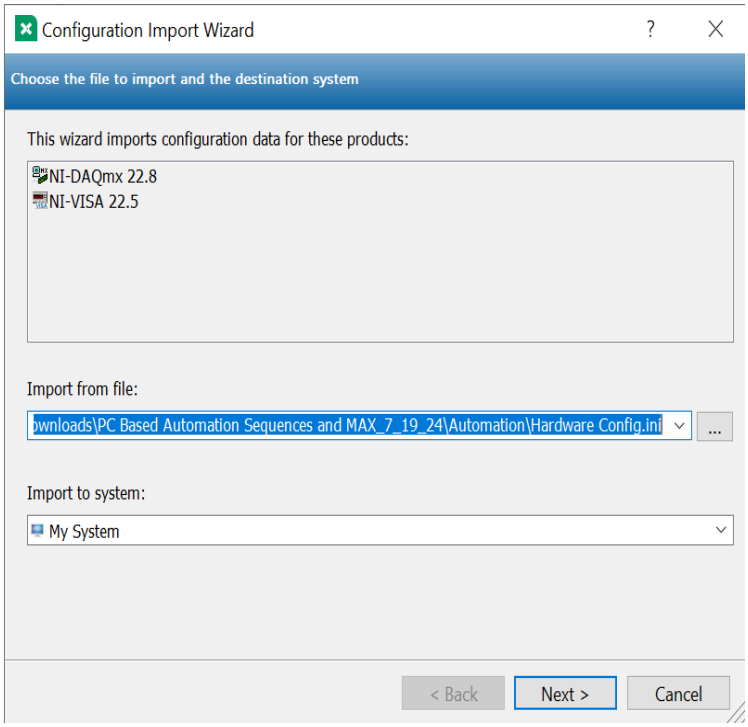
1.5 How to run this Example?

Complete the following steps to run the sequence.

1. First, we must configure NI-MAX to reflect the virtual channels which will be used by the Python script names mentioned in ***analog_voltage_measurement_test*** and ***analog_pwm_test***.
    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

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

"C:\\Windows\\Temp\\ DRVM_test.txt"
"C:\\Windows\\Temp\\ Analog_PWM_Test.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 ***led_test_main_sequence.py*** along with ***analog_voltage_measurement_test.py*** and ***analog_pwm_test.py*** in the IDE or text editor of choice. The following steps are performed within ***led_test_main_sequence.py***.
    a. **Analog Voltage Measurement Test** - Measures and validates the differential Voltage across Anode and Cathode of LED with Analog input resource. Below are the steps included in the test.
        i. Initialize DC-RMS Voltage Measurement Library by creating instance of ***DcRmsVoltageMeasurement()*** class and then using ***initialize()*** method on each object.
        ii. Configure the DC-RMS Voltage Measurement in differential mode and measure the differential voltage across Anode and Cathode.
        iii. Fetch the Voltage measured and returns the measurement.
        iv. Use the ***close()*** methods on both instances to stop generation and measurement.

    b. **Analog PWM Test** (Analog Buffer) - PWM signals are sent to LED circuit with Analog Output resource and the PWM response is captured in the TestPoint with Analog Input resource. Hardware Triggers are used to reduce the delay between PWM generation and capture. Below are the steps included in the test.
        i. Initialize Signal Voltage Generation and Time Domain Measurement Library by creating the instances of ***SignalVoltageGeneration()*** and ***TimeDomainMeasurement()*** classes and then use the ***initialize()*** method on each object.
        ii. Configure the Time Domain Measurement library to wait for the trigger from "Signal Voltage Generation" which sends PWM signal.
        iii. Generate the PWM Signal of configured frequency and Duty cycle with Signal Voltage generation library. Here the example generates **a signal between 0 and 1V at 80% duty cycle**. (in the backend, Analog Output resource sends the Trigger in the backplane which in turn starts the PWM measurement in Analog Input resource)
        iv. Measure the sourced DC Voltage with ***TimeDomainMeasurement()*** class instance by calling the ***configure_and_measure()*** method.
        v. Use the ***close()*** methods on both instances to stop generation and measurement.

    c. **Turn Off all AO Channels** – Power downs all Analog output channels by configuring them to 0 volts. Below are the steps included in the test.
        i. Initialize the DC Voltage Generation library by creating an instance of ***DcVoltageGeneration()*** class and then using ***Initialize()*** method.

ii. Configure the DC Voltage Generation to source 0 Volts in specified Analog Output channels by calling the ***configure_and_generate()*** method using the default parameters.
iii. Use ***close()*** instance after setting AO channels to 0 Volts.

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?

LED 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 "Import Hardware Config" in **section 1.5.** This step imports the Simulated Hardware and creates Global Virtual Channels with Simulated instrument in NI-MAX.
2. Follow the below steps for each sequence. Refer "**Note to run with Hardware**" labels in the sequence.
    i. ***Analog Voltage Measurement Test***
        a. Step into the "***analog_voltage_measurement_test.py***" file.
        b. Update the *"INPUT_TERMINAL"* input with DC-Rms Voltage Measurement Resource Channel in the ***initialize()*** method of ***DcRmsVoltageMeasurement()***.
        c. Update the physical channel name.
        d. Update the "*digital_start_trigger_source*" based on NI-MAX Hardware in the *"DigitalStartTriggerParameters*" configured in the ***configure_and_measure()*** method.
        e. Ensure the *"sample_timing_engine"* configured in *"SampleClockTimingParameters"* Matches with Trigger line defined in previous step
        f. Set the *"MeasurementAnalysisRequirement"* input to *"PROCEED_TO_ANALYSIS"* in *"MeasurementOptions"* property to perform dc-rms voltage analysis in the measured data
        g. Review the configuration of Analog input pins for the intended use case.

    ii. ***Analog PWM Test (Analog Buffer)***
        a. Step into the "***Analog_pwm_test.py***" example.
        b. Update the physical channel name in the "INPUT TERMINAL" and "OUTPUT_TERMINAL.
        c. Update the "Digital Trigger Source" based on Analog Output resource.

d. Review the Configurations of Analog Output Pins and Analog Input Pins for the intended use case

iii. ***Turn Off all AO Channels***

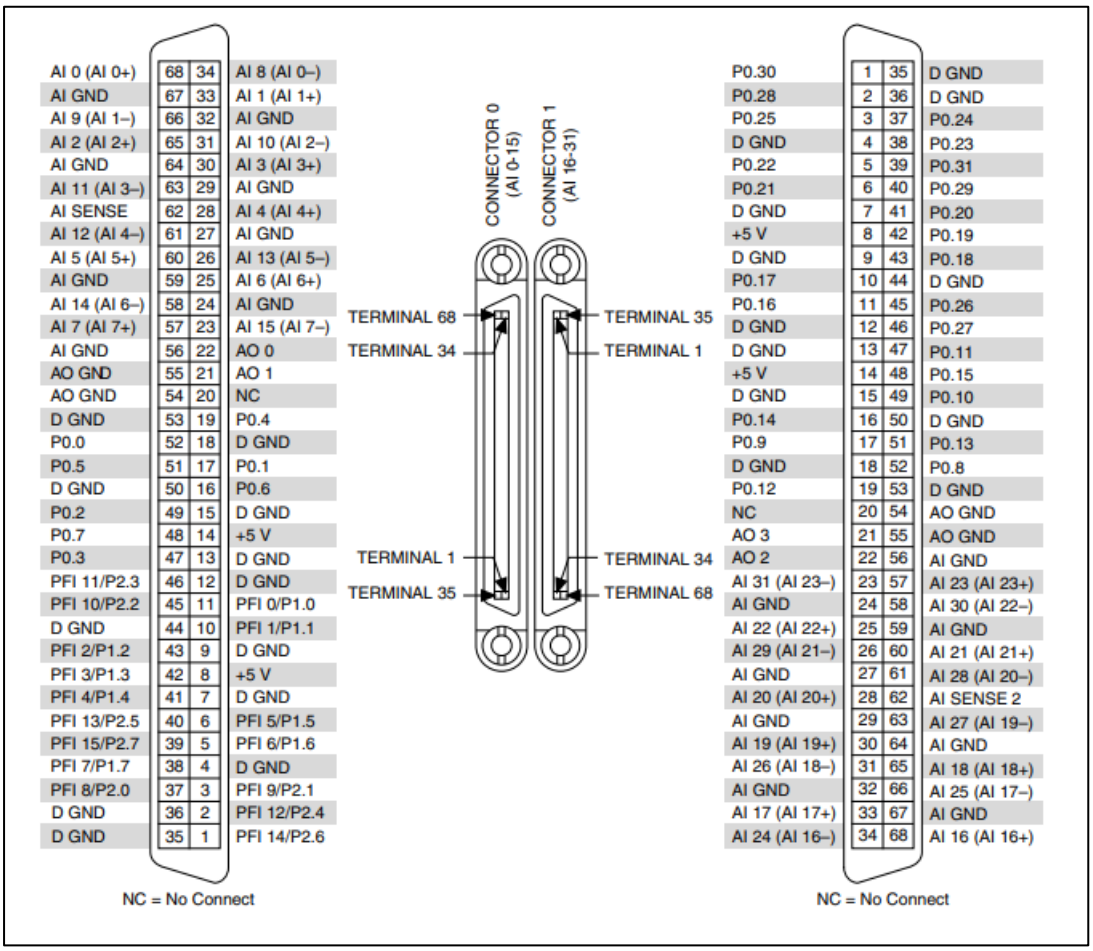
a. Step into the "***turn_off_all_ao_channels.py***" example.

b. Update the physical channels input with Analog Output Channel in the initialize step of ***Turn_off_all_ao_channels.***

c. Review the Configurations of Analog Output for the intended use case

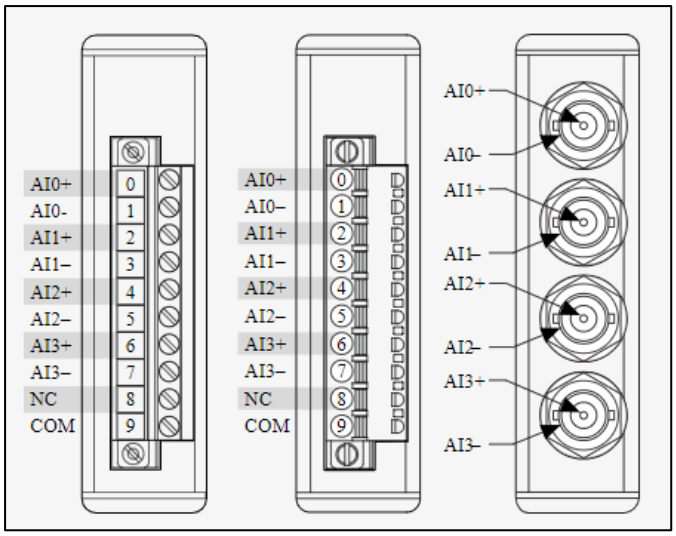
3. Review the **generate and range settings of Analog Output Pins and Analog input Pins** based on the DUT and Connections before running with Hardware.

1.6 Pinouts of PCIe-6323

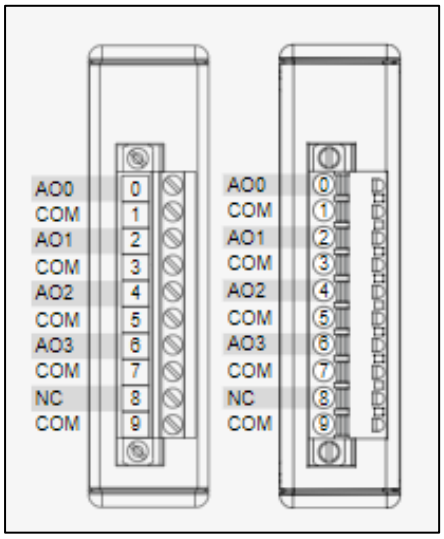


1.7 Pinouts of NI-9215 AI Module

1. **Analog Input Module (NI-9215)**

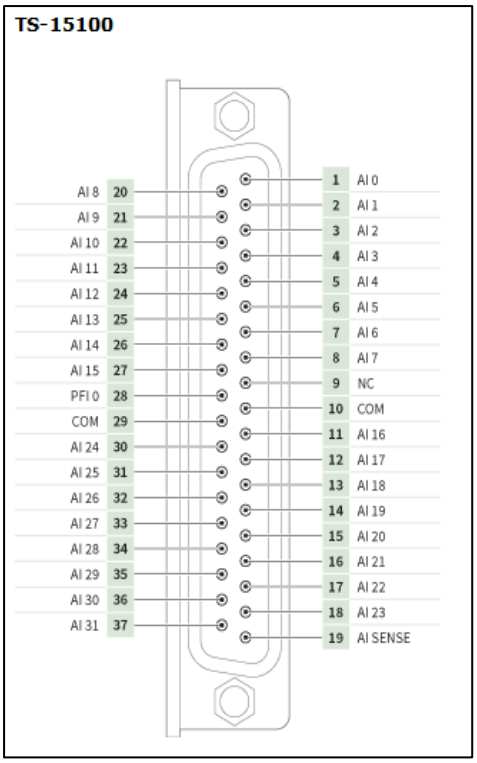


2. **Analog Output Module (NI-9263)**

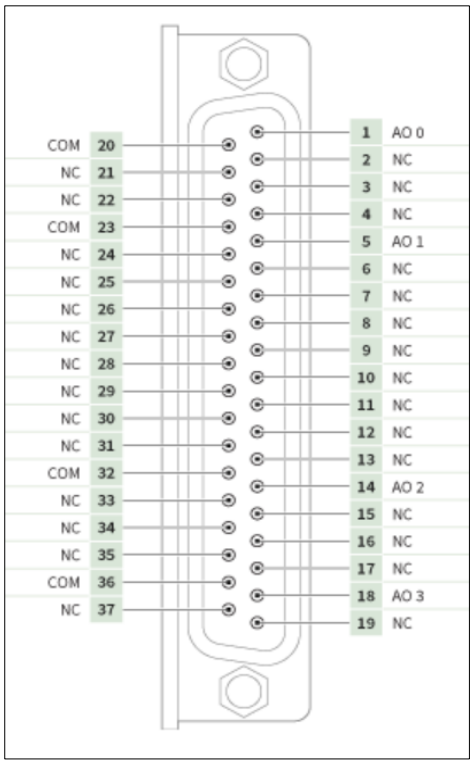


1.8 Pinouts of TestScale Modules

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



2. **Analog Output Module (TS-15110)**



1.9 How to create/Modify Global Virtual Channels?

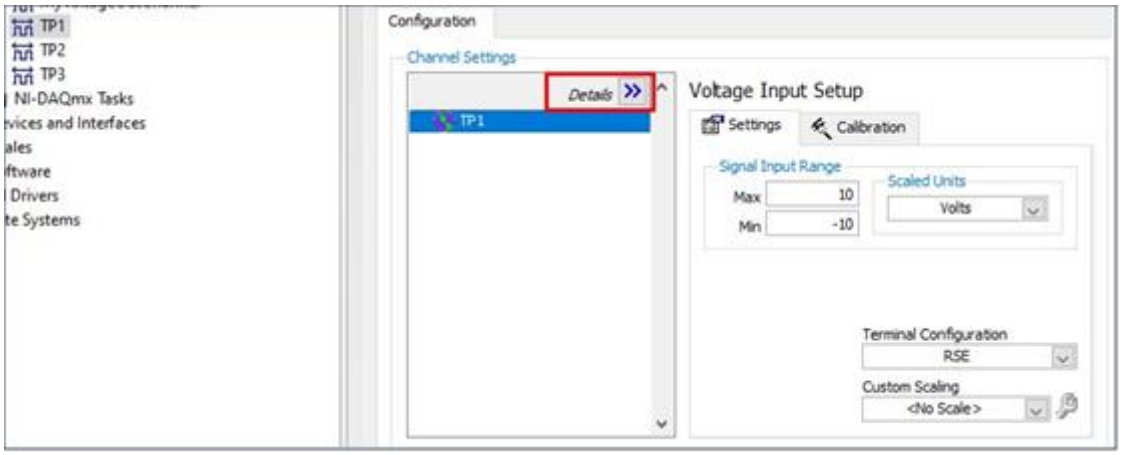
A virtual channel is a collection of settings such as a name, a physical channel, input terminal connections, the type of measurement or generation, and can include scaling information. A virtual channel created outside a task is a Global Virtual Channel.

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

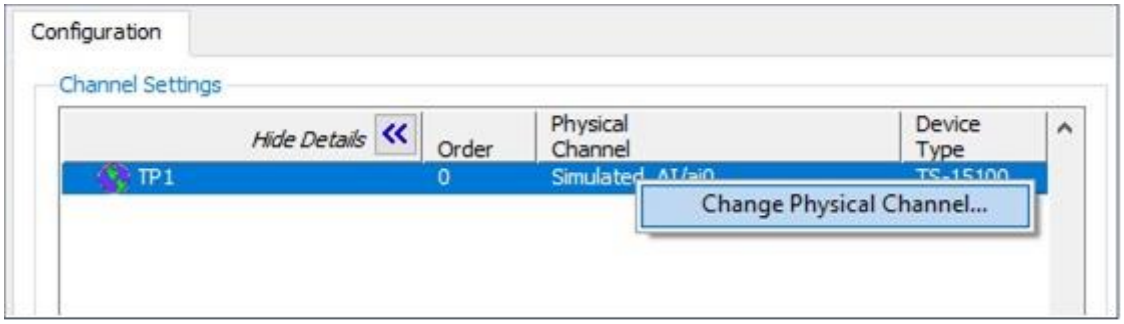
1. Launch NI-MAX
2. In NI-MAX, right-click **Data Neighbourhood** and select **Create New**
3. In the Create New window, select **NI-DAQmx Global Virtual Channel** and click **Next**. The DAQ Assistant opens.
4. Select an I/O type, such as analog input
5. Select the physical channel of Hardware 6
6.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