Device Synchronization Example Sequence

1.1 Purpose

Device Synchronization Example sequence demonstrate multiple device synchronization using Generation and Measurement Library targeting both Analog and Digital IO resources. This example sequence can be executed in a custom Python sequence script using the measurement libraries written in Python.

Example File Location

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

1.2 Highlighted Features

- Signal Voltage Generation and Time Domain Measurement
  - Supplies analog sine tones through Analog Ouput resources and performs Time domain measurements on the captured Analog Input waveforms by Analog Input resources. Sample clock and Hardware Triggers are shared between Source and Measure to achieve Synchronization. Libraries used in the example are "***SynchronizationSignalRouting()"***, "***SignalVoltageGeneration()"*** and ***"TimeDomainMeasurement()***".
- Dynamic Digital Pattern Generation and Dynamic Digital Pattern Measurement
  - Generates Digital Pattern using Digital output resources and measure the same using digital input resources. Sample clock and Hardware Triggers are shared between Source and Measure to achieve Synchronization. Libraries used in the example are "***SynchronizationSignalRouting()"***, "***DynamicDigitalPatternGeneration()"*** and "***DynamicDigitalPatternMeasurement()"***.
- Digital Clock Generation and Digital PWM Measurement
  - Generates Digital Clock Signal using Digital output resources and measure the same using digital input resources to obtain PWM Measurements. No Sample clock or Hardware Triggers are shared among libraries in this example. Digital Input resource is armed to capture Digital PWM Signals before generation start. Libraries used in the example are "***DigitalClockGeneration()"*** and "***DigitalPwmMeasurement()"***.
- 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()".***
- Turn Off all DO Channels
  - Powers down all digital output channels by configuring the output state to LOW. Libraries used in the example is "**StaticDigitalStateGeneration()".**

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

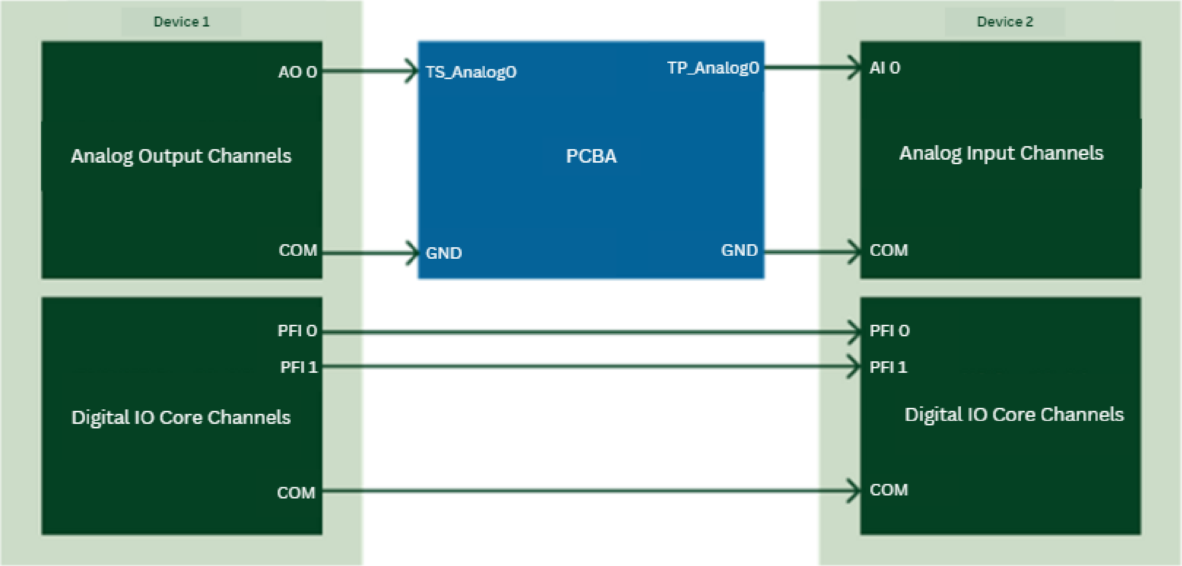
1.3 Prerequisites

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

1.4 Setup Diagram

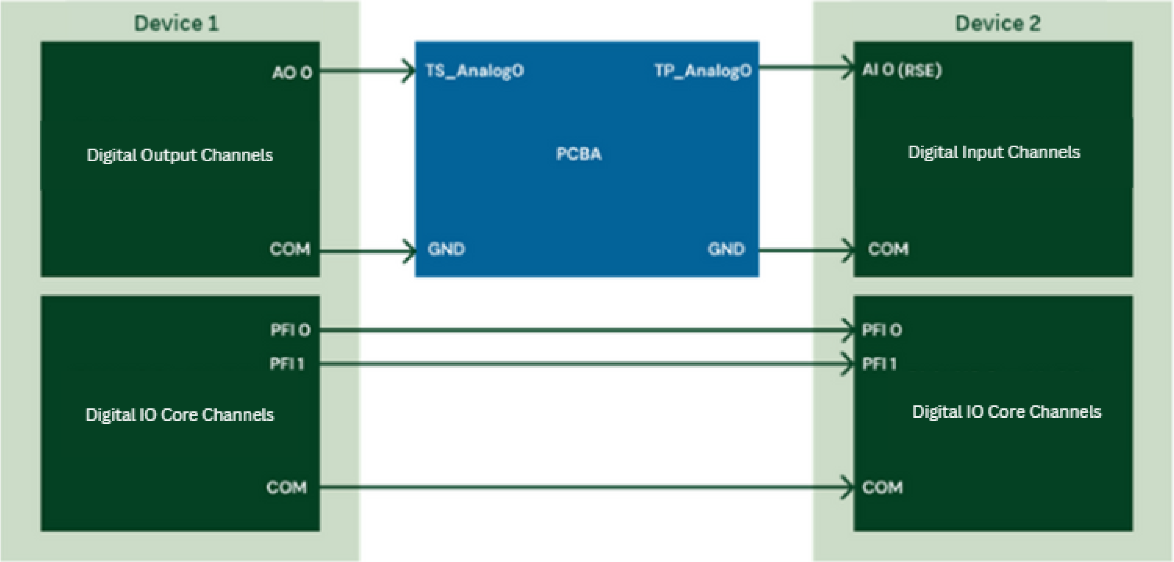
Represents the hardware setups used in this example sequence. Pin Outs of each resource is added below.

1. Setup A – Signal Voltage Generation and Time Domain Measurement
   PFI0 and PFI1 is used to route Sample Clock and Start Trigger respectively.

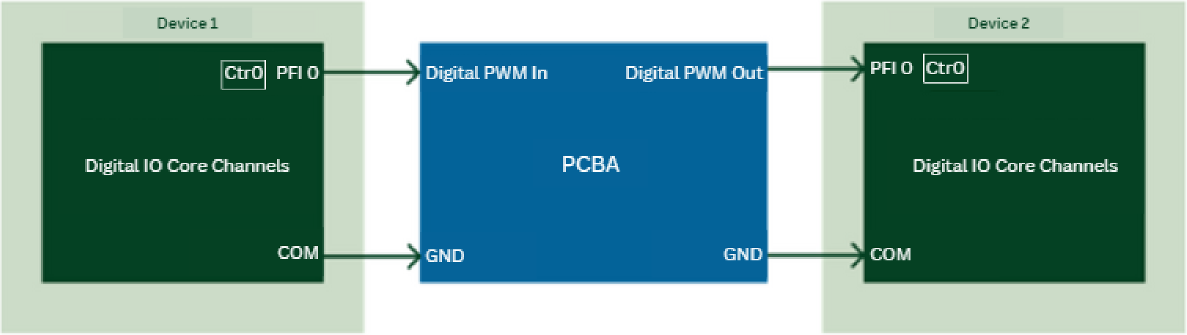


2. Setup B – Dynamic Digital Pattern Generation and Dynamic Digital Pattern Measurement

   PFI0 and PFI1 is used to route Sample Clock and Start Trigger respectively.



3. Setup C – Digital Clock Generation and Digital PWM Measurement



1.5 How to run this Example?

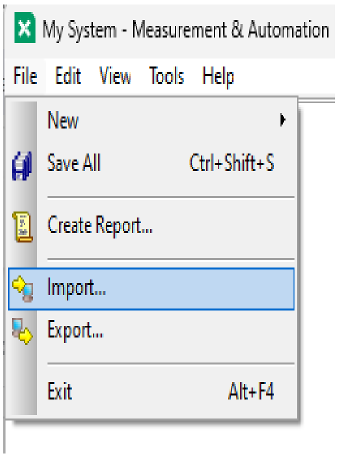
Complete the following steps to run the sequence.

1. First, we must configure NI-MAX to reflect the simulated virtual channels which will be used by the Python script names mentioned in ***digital_clock_generation_and_pwm_measurement.py*** or ***dynamic_digital_pattern_generation_and_measurement.py*** or ***signal_voltage_and_time_domain_measure.py :***

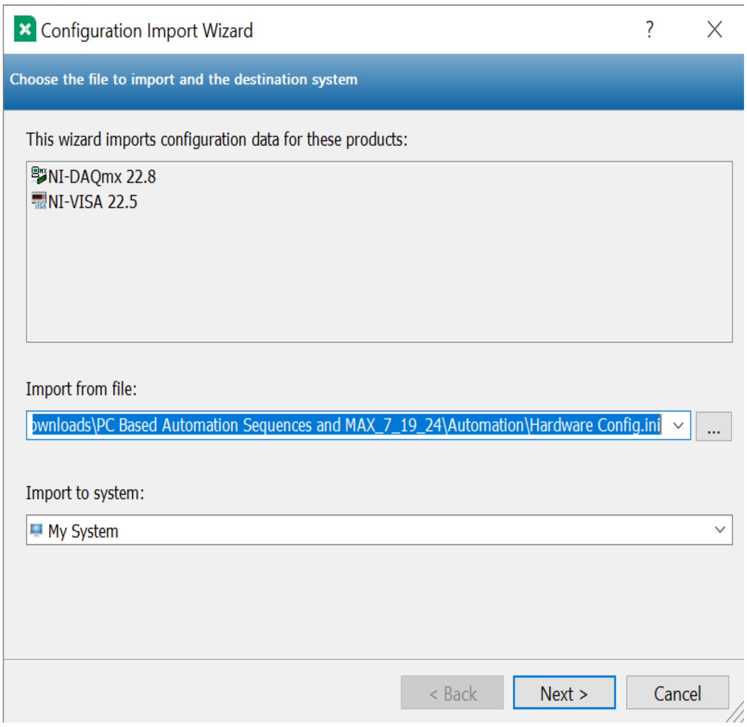
a. A hardware configuration file for NI-MAX is required to run this example. The configuration file contains a set of predefined global channel names which are used by the nidaqmx driver to communicate with the Python scripts.

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*



Choose the file to import and the destination system

This wizard imports configuration data for these products:

NI-DAQmx 22.8
NI-VISA 22.5

Import from file: ownloads\PC Based Automation Sequences and MAX_7_19_24\Automation\Hardware Config.ini

Import to system: My System

e. NI-MAX now holds the same virtual channel name references contained in the examples provided

The ***digital_clock_generation_and_pwm_measurement.py, dynamic_digital_pattern_generation_and_measurement.py and signal_voltage_and_time_domain_measure.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\\digital_clock_generation_and_pwm_measurement_results.txt"*
> *"C:\\Windows\\Temp\\dynamic_digital_pattern_generation_and_measurement_results.txt"*
> *"C:\\Windows\\Temp\\signal_voltage_and_time_domain_measurement_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 ***synchronization_main_sequence.py*** *along with* ***digital_clock_generation_and_pwm_measurement.py***, ***dynamic_digital_pattern_generation_and_measurement.py and signal_voltage_and_time_domain_measure.py*** in your IDE or text editor of choice. The following steps are performed within ***synchronization_main_sequence.py.***
    a. **Signal Voltage and Time Domain Measure** - Supplies analog sine tones through Analog Output resources (in device 1) and performs Time domain measurements on the captured Analog Input waveforms (in device 2) by Analog Input resources. Sample clock and Hardware Triggers are shared through PFI lines present in DIO Core resource to achieve Synchronization. Below are the steps included in the test.
        i. Initialize Signal Voltage Generation for AO resource in Device 1 and Time Domain Measurement Libraries for AI resource in Device 2 by creating instances of ***SignalVoltageGeneration()*** and ***TimeDomainMeasurement()*** then using ***initialize()*** method on each object.

ii. Route to export the Sample clock and Start trigger signals to DIO Core resource PFI lines using Synchronization Library for Signal Voltage Generation Task in Device 1.

iii. Configure Time Domain Measurement to wait for Start Trigger and Sample clock from Signal Voltage Generation through PFI Lines.
Note: The PFI lines are connected physically between Device 1 and 2.

iv. Configure the Signal Voltage Generation to start sourcing Sine Tones (under the hood, Signal Voltage Generation in Device 1 resource sends the Trigger and Clock through the PFI lines externally during sourcing starts which in-turns starts the measurement in Analog Input resource in Device 2). In this example, a Single tone sine waveform with frequency of **100Hz** and amplitude of **1 Volt** is generated for a duration of **0.1 seconds**.

v. Fetches the captured single tone sine waveforms with ***TimeDomainMeasurement()*** class instance by calling ***configure_and_measure()*** method and return the time measurements.

vi. 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. **Dynamic Digital Pattern Generation and Measurement** - Generates Digital Pattern using Digital output resources (in Device 1) and measure the same using digital input resources (in Device 2).
Sample clock and Hardware Triggers are shared through PFI lines present in DIO Core resource to achieve Synchronization. Below are the steps included in the test,

i. Initialize Dynamic Digital Pattern Generation for DO resource in Device 1 and Dynamic Digital Pattern Measurement Libraries for DI resource in Device 2 by creating the instances of **DynamicDigitalPatternGeneration()** and ***DynamicDigitalPatternMeasurement()*** classes and then using ***initialize()*** methods on each object.

ii. Route to export the Sample clock and Start trigger signals to DIO Core resource PFI lines using Synchronization Library for Dynamic Digital Pattern Generation Task in Device 1.

iii. Configure Dynamic Digital Pattern Measurement to wait for Start Trigger and Sample clock from Dynamic Digital Pattern Generation through PFI Lines.
Note: The PFI lines are connected physically between Device 1 and 2.

iv. Configure the Dynamic Digital Pattern Generation to start sourcing Digital Pattern (under the hood, Dynamic Digital Pattern Generation in Device 1 resource sends the Trigger and Clock through the PFI lines externally during sourcing starts which in-turns starts the measurement in Digital Input resource in Device 2). In this example, a Digital Ramp Pattern Generator method is used to generate ramp based digital pattern with 1000 samples for two Digital Output lines.

v. Fetches the captured Digital Pattern with ***DynamicDigitalPatternMeasurement()*** class instance by calling the method ***configure_and_measure()*** and displays in Port Digital Data format.

vi. 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.

c. **Digital Clock Generation and PWM Measurement** - Generates Digital Clock Signal using Digital output resources (in Device 1) and measure the same using digital input resources (in Device 2) to obtain PWM Measurements. No Sample clock or Hardware Triggers are shared among libraries to achieve Synchronization. But Digital Input resource (in Device 2) is armed to capture Digital PWM Signals before generation start. Below are the steps included in the test,

   i. Initialize Digital Clock Generation for Core Resource of Device 1 and Digital PWM Measurement Libraries for Core Resource of Device 2 by creating instances of ***DigitalClockGeneration()*** and ***DigitalPwmMeasurement()*** classes and then using ***initialize()*** method on each object.

   ii. Configure Digital PWM Measurement Library to wait for detection of PWM signals and Number of Cycles to capture. The Number of Cycles is configured to capture **100** PWM cycles.

   iii. Configure the Digital Clock Generation and start producing **pulse train of 1kHz** with **50%** Duty Cycle for **0.1** second through PFI Lines.
   Note: PWM Measurement library starts the capture at the first rising/falling edge of the PWM signal through PFI lines.

   iv. Fetch the PWM Measurement measured by the Digital Input Resource in Device 2 using ***DigitalPwmMeasurement()*** class instance by calling the ***configure_and_measure()*** method.

   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

d. **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.

e. **Turn Off all DO Channels** – Power downs all Digital output channels by configuring them to LOW state. Below are the steps included in the test.

   i. Initialize Static Digital State Generation Library by creating an instance of ***StaticDigitalStateGeneration()*** class and then using ***Initialize()*** method

   ii. Configure the Static Digital State Generation to source state LOW in specified Digital Output channels by calling the ***configure_and_generate()*** method using the default parameters.

   iii. Use ***close()*** instance after setting DO channels to ***False***.

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 run with Hardware?

Synchronization Demo 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. ***Signal Voltage and Time Domain Measure***
        1. Step into the "***signal_voltage_and_time_domain_measure"*** sequence.
        2. Open NI-MAX and update the physical Channel linked to the Global Channels – **TS_Analog0, TP_Analog0** (called in the initialize step of Signal Voltage Generation and Time Domain Measurement)
        3. Update the "routing path" in "Route Synchronization Signals" step for Device 1, to export Sample clock and Start Trigger of Signal Voltage Generation task to the targeted PFI lines based on Setup connection.
        4. Update the "Sample Clock Source" and "Digital Trigger Source" to external PFI lines in Device 2 based on Setup connection in the "Time Domain Measurement – Configure TP" step.
        Note: Refer the How to achieve synchronization? Topic for more details.
        5. Update "Periodic Waveform" input to True to enable all Time Domain Measurements in "Time Domain Measurement - Read Captured Waveform" step.
        6. Review the Configurations of Analog Output and Analog Input Pins for the intended use case
    ii. ***Dynamic Digital Pattern Generation and Measurement***
        1. Step into the "***dynamic_digital_pattern_generation_and_measurement"*** sequence***.***
        2. Open NI-MAX and update the physical Channel linked to the Global Channels – **TS_Digital0:1, TP_ Digital0:1** (called in the initialize step of Dynamic Digital Pattern Generation and Measurement)

3. Update the routing path in Device 1, to export Sample clock and Start Trigger of Dynamic Digital Pattern Generation task to the targeted PFI lines based on Setup connection.
4. Update the "Sample Clock Source" and "Digital Trigger Source" to external PFI lines in TestScale Device 2 based on Setup connection in the "Dynamic Digital Pattern Measurement - Configure TP" step.
   Note: Refer the How to achieve synchronization? Topic for more details.
5. Update the Number of Digital Lines configured for generation in Generate Port Digital Data step.
6. Review the Configurations of Digital Output and Digital Input Pins for the intended use case.

iii. ***Digital Clock Generation and PWM Measurement***

1. Step into the "***digital_clock_generation_and_pwm_measurement***" sequence
2. Update Output Terminal and Physical Channel (Counter) in "Digital Clock Generation – Initialize" and "Digital PWM Measurement – Initialize" step based on Setup connection.
   Note: Refer the How to achieve synchronization? Topic for more details.
3. Review the Configurations of Digital Output and Digital Input Pins for the intended use case

iv. ***Turn off all AO Channels***

1. Step into the "***turn_off_all_ao_channels***" sequence
2. Update the "Physical Channels" input with Analog Output Channel in the initialize step of Turn Off All AO Channels sequence
3. Review the Configurations of Analog Output for the intended use case

v. ***Turn Off all DO Channels***

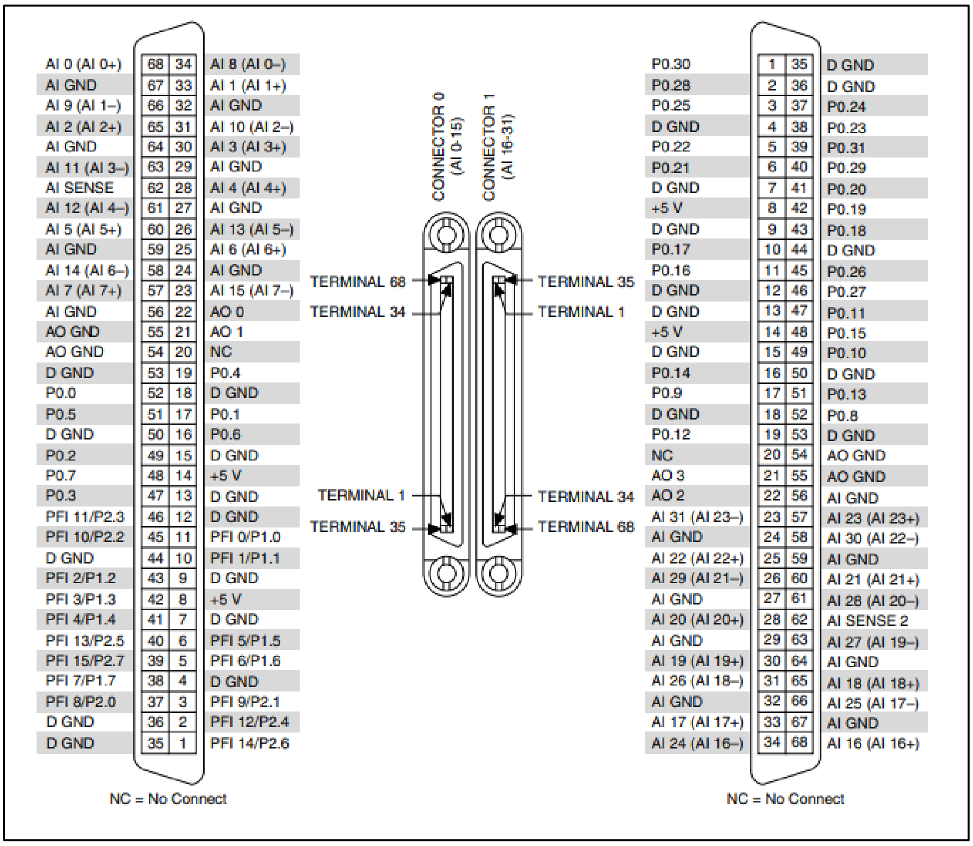
1. Step into the "***turn_off_all_do_channels***" sequence.
2. Update the "Global Channels" input with Digital Output Channels used in the initialize step of Turn Off All DO Channels sequence.
3. Review and Update "Number of Digital Lines" to be configured to Digital False state in "Assign Digital States" Step.
4. Review the Configurations of Digital Output Pins for the intended use case.

1.5.2 How to achieve synchronization?

To achieve better synchronization between multiple devices, NI Suggests following the below standards to get optimum results,

1. During multi-devices synchronization, make sure the external cables used to route signals like Sample clock and Start Trigger between chassis are length matched. This prevents signal skew between trigger and clock.
2. Configure the generation and measurement ends to function at same sample clock rate.
3. During multi-devices synchronization for Digital Signals, Generation of Data in T1 system and Measurement of Data in T2 system are performed with same shared clock signal, there are chances of incorrect data measurements due to metastability caused by setup and hold time violation. To prevent such scenario, NI recommends the following configurations,
    i. In Measurement end, set the 'Active Edge' parameter in Timing settings as 'Falling Edge' and in Generation end, set the same parameter as 'Rising Edge'. By doing so, an offset of ½ Sampling Time Period is added to measurement end to prevent setup and hold violations.
4. For more info on Synchronization, refer Synchronization Explained topic in NI Website.
5. To choose your custom synchronization technique for cDAQ/TestScale Devices, refer Choosing a CompactDAQ Synchronization Technology topic in NI Website.

1.6 Pinouts of PCIe-6323



AI 0 (AI 0+)	68	34	AI 8 (AI 0–)
AI GND	67	33	AI 1 (AI 1+)
AI 9 (AI 1–)	66	32	AI GND
AI 2 (AI 2+)	65	31	AI 10 (AI 2–)
AI GND	64	30	AI 3 (AI 3+)
AI 11 (AI 3–)	63	29	AI GND
AI SENSE	62	28	AI 4 (AI 4+)
AI 12 (AI 4–)	61	27	AI GND
AI 5 (AI 5+)	60	26	AI 13 (AI 5–)
AI GND	59	25	AI 6 (AI 6+)
AI 14 (AI 6–)	58	24	AI GND
AI 7 (AI 7+)	57	23	AI 15 (AI 7–)
AI GND	56	22	AO 0
AO GND	55	21	AO 1
AO GND	54	20	NC
D GND	53	19	P0.4
P0.0	52	18	D GND
P0.5	51	17	P0.1
D GND	50	16	P0.6
P0.2	49	15	D GND
P0.7	48	14	+5 V
P0.3	47	13	D GND
PFI 11/P2.3	46	12	D GND
PFI 10/P2.2	45	11	PFI 0/P1.0
D GND	44	10	PFI 1/P1.1
PFI 2/P1.2	43	9	D GND
PFI 3/P1.3	42	8	+5 V
PFI 4/P1.4	41	7	D GND
PFI 13/P2.5	40	6	PFI 5/P1.5
PFI 15/P2.7	39	5	PFI 6/P1.6
PFI 7/P1.7	38	4	D GND
PFI 8/P2.0	37	3	PFI 9/P2.1
D GND	36	2	PFI 12/P2.4
D GND	35	1	PFI 14/P2.6

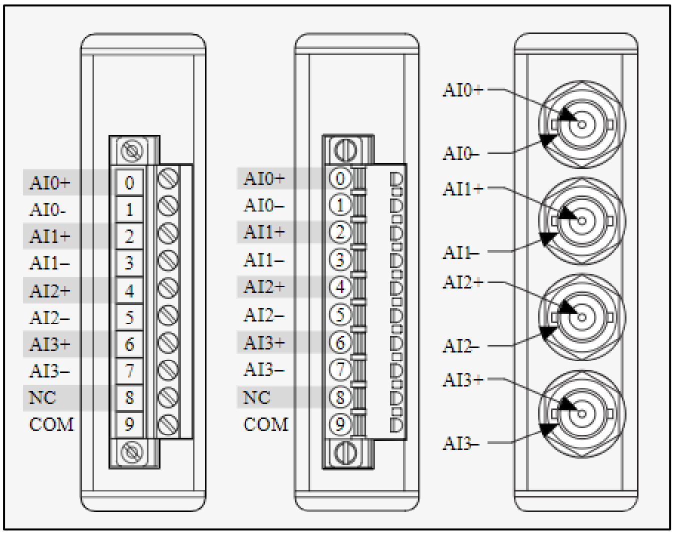
NC = No Connect

P0.30	1	35	D GND
P0.28	2	36	D GND
P0.25	3	37	P0.24
D GND	4	38	P0.23
P0.22	5	39	P0.31
P0.21	6	40	P0.29
D GND	7	41	P0.20
+5 V	8	42	P0.19
D GND	9	43	P0.18
P0.17	10	44	D GND
P0.16	11	45	P0.26
D GND	12	46	P0.27
D GND	13	47	P0.11
+5 V	14	48	P0.15
D GND	15	49	P0.10
P0.14	16	50	D GND
P0.9	17	51	P0.13
D GND	18	52	P0.8
P0.12	19	53	D GND
NC	20	54	AO GND
AO 3	21	55	AO GND
AO 2	22	56	AI GND
AI 31 (AI 23–)	23	57	AI 23 (AI 23+)
AI GND	24	58	AI 30 (AI 22–)
AI 22 (AI 22+)	25	59	AI GND
AI 29 (AI 21–)	26	60	AI 21 (AI 21+)
AI GND	27	61	AI 28 (AI 20–)
AI 20 (AI 20+)	28	62	AI SENSE 2
AI GND	29	63	AI 27 (AI 19–)
AI 19 (AI 19+)	30	64	AI GND
AI 26 (AI 18–)	31	65	AI 18 (AI 18+)
AI GND	32	66	AI 25 (AI 17–)
AI 17 (AI 17+)	33	67	AI GND
AI 24 (AI 16–)	34	68	AI 16 (AI 16+)

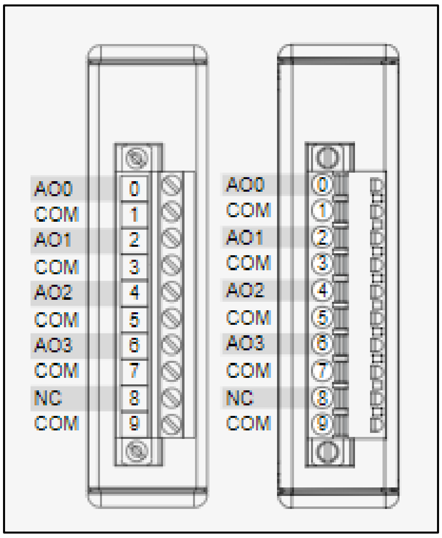
NC = No Connect

1.7 Pinouts of cDAQ Modules

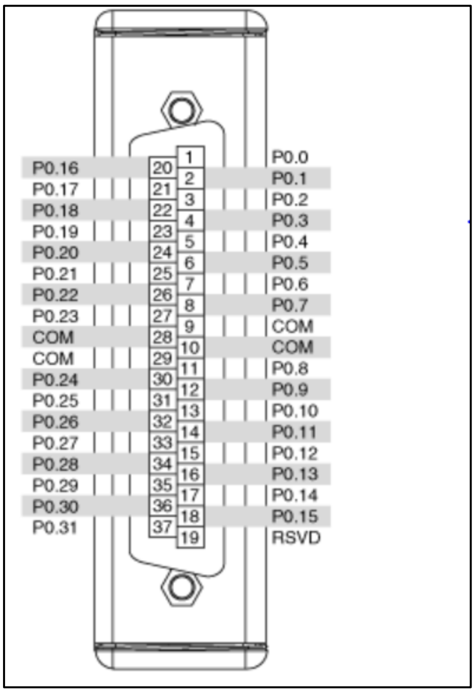
1. **Analog Input Module (NI-9215)**



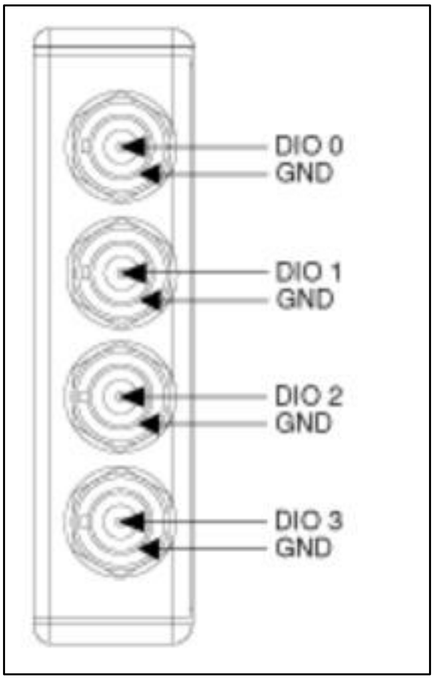
2. **Analog Output Module (NI-9263)**



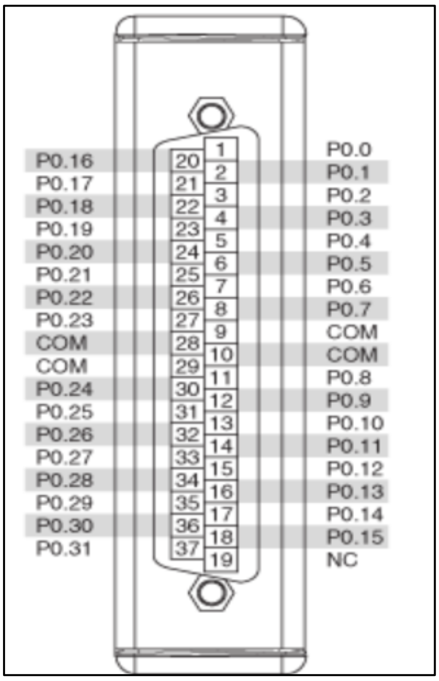
3. **Digital Input Output Module (NI-9403)**



4. **PFI Module (NI-9402)**

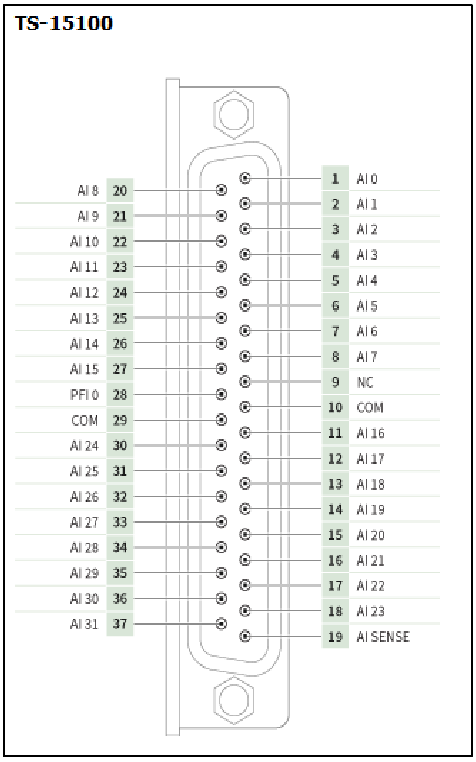


3. **Digital Output Module (NI-9477)**

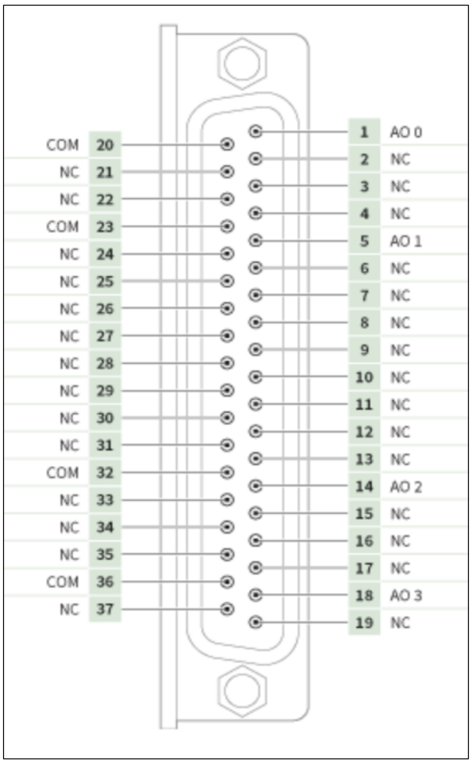


1.8 Pinouts of TestScale Modules

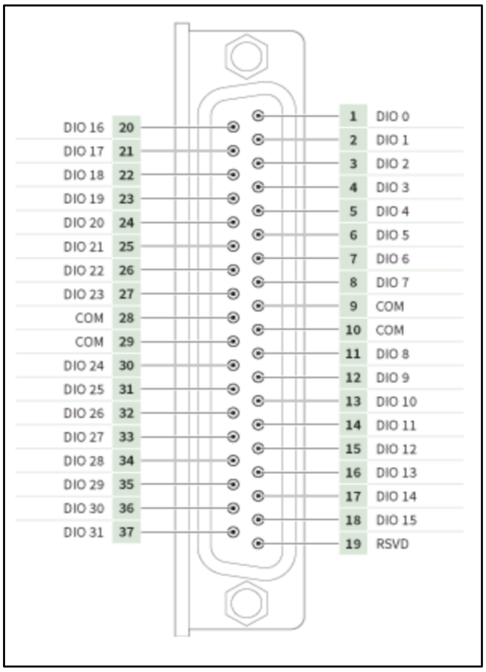
1. **Analog Input Module (TS-15100)**



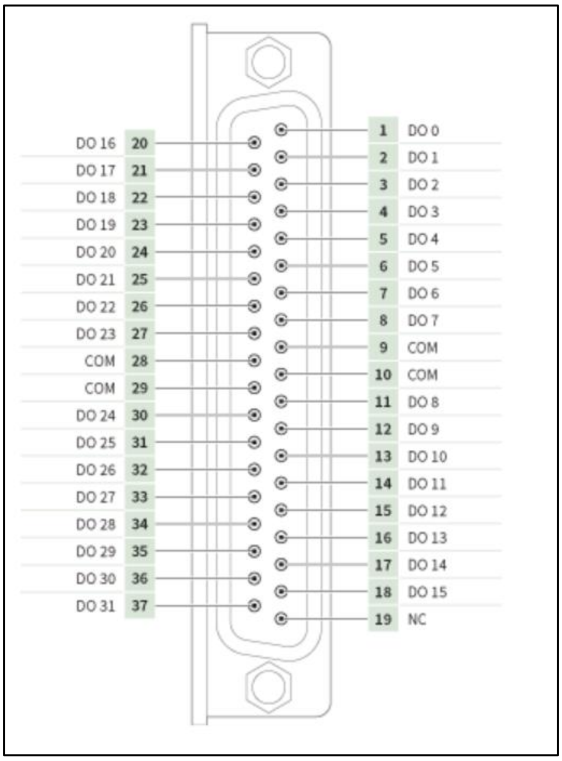
2. **Analog Output Module (TS-15110)**



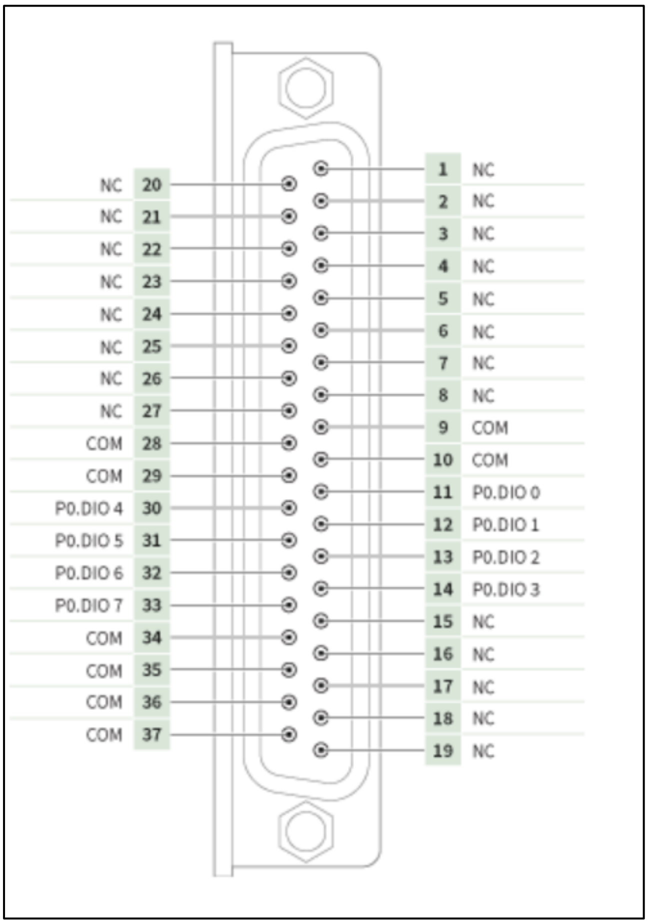
3. **Digital Input Output Module (TS-15120)**



4. **Digital Output Module (TS-15110)**



3. **Core DIO Module (TS-15050)**



1.9 How to create/Modify Global Virtual Channels?

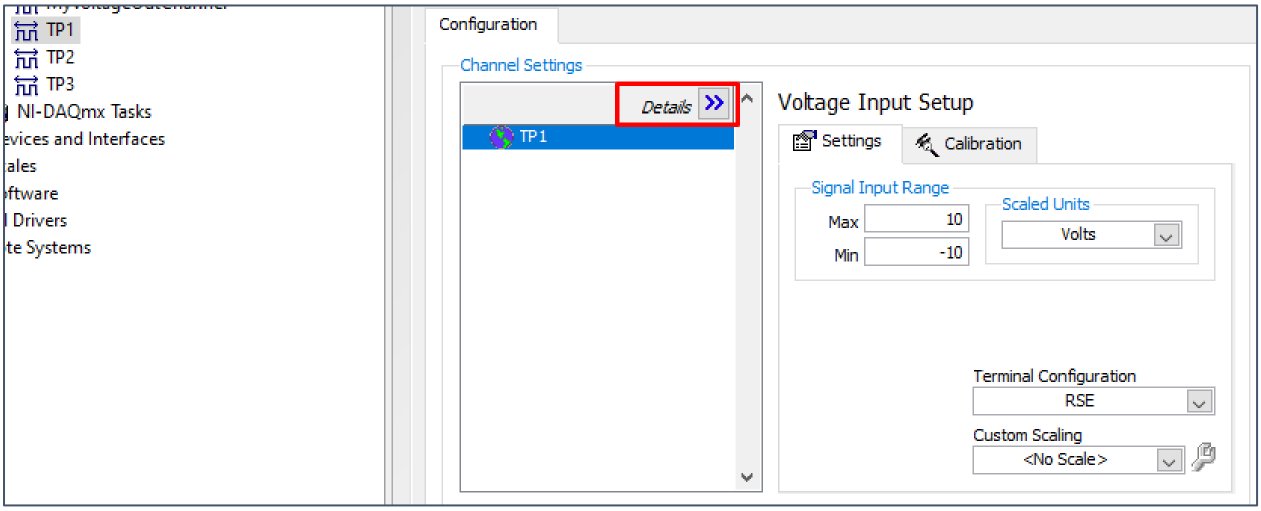
A virtual channel is a collection of settings such as a name, a physical channel, input terminal connections, the type of measurement or generation, and can include scaling information. A virtual channel created outside a task is a Global Virtual Channel. Follow the below steps to **create Global Virtual Channel** in NI-MAX.

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**
4. In the Create New window, select **NI-DAQmx Global Virtual Channel** and click **Next**.
5. 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.

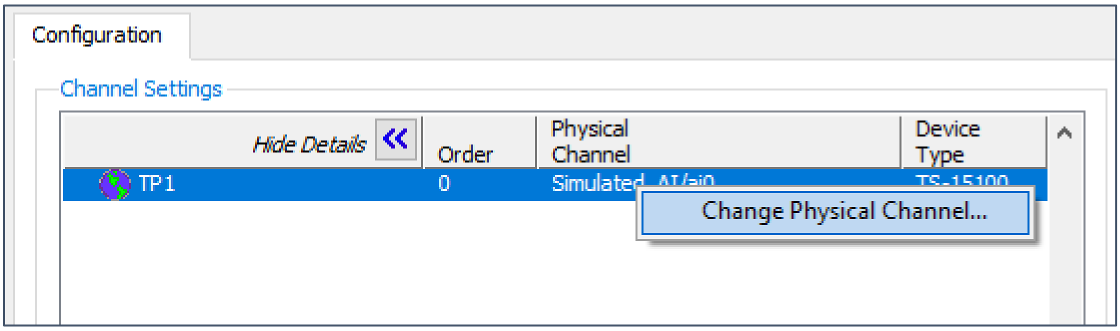
Follow the below steps to **modify the existing Global Virtual Channel** in NI-MAX.

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



Max 10
Min -10
Scaled Units: Volts
Terminal Configuration: RSE
Custom Scaling: <No Scale>

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”



	Order	Physical Channel	Device Type
TP1	0	Simulated_AI/ai0	TS-15100

6. **Save** your configuration