NI-TClk Python API Documentation

Release 1.4.9.dev0

NI

May 01, 2024

DOCUMENTATION

CHAPTER

ONE

ABOUT

The **nitclk** module provides a Python API for NI-TClk. The code is maintained in the Open Source repository for nimi-python.

1.1 Support Policy

nitclk supports all the Operating Systems supported by NI-TClk.

It follows Python Software Foundation support policy for different versions of CPython.

CHAPTER

TWO

CONTRIBUTING

We welcome contributions! You can clone the project repository, build it, and install it by following these instructions.

CHAPTER

THREE

SUPPORT / FEEDBACK

For support specific to the Python API, follow the processs in *Bugs / Feature Requests*. For support with hardware, the driver runtime or any other questions not specific to the Python API, please visit NI Community Forums.

CHAPTER

FOUR

BUGS / FEATURE REQUESTS

To report a bug or submit a feature request specific to Python API, please use the GitHub issues page.

Fill in the issue template as completely as possible and we will respond as soon as we can.

4.1 nitclk module

4.1.1 Installation

As a prerequisite to using the **nitclk** module, you must install the NI-TClk runtime on your system. Visit ni.com/downloads to download the driver runtime for your devices.

The nimi-python modules (i.e. for **NI-TClk**) can be installed with pip:

```
$ python -m pip install nitclk
```

4.1.2 Usage

The following is a basic example of using the **nitclk** module

```
import nitclk
```

Other usage examples can be found on GitHub.

4.1.3 API Reference

Public API

The *nitclk* module provides synchronization facilites to allow multiple instruments to simultaneously respond to triggers, to align Sample Clocks on multiple instruments, and/or to simultaneously start multiple instruments.

It consists of a set of functions that act on a list of *SessionReference* objects or instrument *Session* objects for drivers that support NI-TClk. *SessionReference* also has a set of properties for configuration.

```
with niscope.Session('dev1') as scope1, niscope.Session('dev2') as scope2:
    nitclk.configure_for_homogeneous_triggers([scope1, scope2])
    nitclk.initiate([scope1, scope2])
    wfm1 = scope1.fetch()
    wfm2 = scope2.fetch()
```

configure_for_homogeneous_triggers

nitclk.**configure_for_homogeneous_triggers**(*sessions*)

Configures the properties commonly required for the TClk synchronization of device sessions with homogeneous triggers in a single PXI chassis or a single PC. Use *nitclk.configure_for_homogeneous_triggers()* to configure the properties for the reference clocks, start triggers, reference triggers, script triggers, and pause triggers. If *nitclk.configure_for_homogeneous_triggers()* cannot perform all the steps appropriate for the given sessions, it returns an error. If an error is returned, use the instrument driver methods and properties for signal routing, along with the following NI-TClk properties: *nitclk.SessionReference.start_trigger_master_session* *nitclk.SessionReference.ref_trigger_master_session* *nitclk.SessionReference.pause_trigger_master_session* *nitclk.configure_for_homogeneous_triggers()* affects the following clocks and triggers: - Reference clocks - Start triggers - Reference triggers - Script triggers - Pause triggers Reference Clocks *nitclk.configure_for_homogeneous_triggers()* configures the reference clocks if they are needed. Specifically, if the internal sample clocks or internal sample clock timebases are used, and the reference clock source is not configured–or is set to None (no trigger configured)–*nitclk.configure_for_homogeneous_triggers()* configures the following: PXI–The reference clock source on all devices is set to be the 10 MHz PXI backplane clock (PXI_CLK10). PCI–One of the devices exports its 10 MHz onboard reference clock to RTSI 7. The reference clock source on all devices is set to be RTSI 7. Note: If the reference clock source is set to a value other than None, *nitclk.configure_for_homogeneous_triggers()* cannot configure the reference clock source. Start Triggers If the start trigger is set to None (no trigger configured) for all sessions, the sessions are configured to share the start trigger. The start trigger is shared by: - Implicitly exporting the start trigger from one session - Configuring the other sessions for digital edge start triggers with sources corresponding to the exported start trigger - Setting *nitclk.SessionReference.start_trigger_master_session* to the session that is exporting the trigger for all sessions If the start triggers are None for all except one session, *nitclk.configure_for_homogeneous_triggers()* configures the sessions to share the start trigger from the one excepted session. The start trigger is shared by: - Implicitly exporting start trigger from the session with the start trigger that is not None - Configuring the other sessions for digital-edge start triggers with sources corresponding to the exported start trigger - Setting *nitclk.SessionReference.start_trigger_master_session* to the session that is exporting the trigger for all sessions If start triggers are configured for all sessions, *nitclk.configure_for_homogeneous_triggers()* does not affect the start triggers. Start triggers are considered to be configured for all sessions if either of the following conditions is true: - No session has a start trigger that is None - One session has a start trigger that is None, and all other sessions have start triggers other than None. The one session with the None trigger must have *nitclk.SessionReference.start_trigger_master_session* set to itself, indicating that the session itself is the start trigger master Reference Triggers *nitclk.configure_for_homogeneous_triggers()* configures sessions that support reference triggers to share the reference triggers if the reference triggers are None (no trigger configured) for all except one session. The reference triggers are shared by: - Implicitly exporting the reference trigger from the session whose reference trigger is not None - Configuring the other sessions that support the reference trigger for digital-edge reference triggers with sources corresponding to the exported reference trigger - Setting *nitclk.SessionReference.ref_trigger_master_session* to the session that is exporting the trigger for all sessions that support reference trigger If the reference triggers are configured for all sessions that support reference triggers, *nitclk.configure_for_homogeneous_triggers()* does not affect the reference triggers. Reference triggers are considered to be configured for all sessions if either one or the other of the following conditions is true: - No session has a reference trigger that is None - One session has a reference trigger that is None, and all other sessions have reference triggers other than None. The one session with the None trigger must have *nitclk.SessionReference.ref_trigger_master_session* set to

itself, indicating that the session itself is the reference trigger master Reference Trigger Holdoffs Acquisition sessions may be configured with the reference trigger. For acquisition sessions, when the reference trigger is shared, *nitclk.configure_for_homogeneous_triggers()* configures the holdoff properties (which are instrument driver specific) on the reference trigger master session so that the session does not recognize the reference trigger before the other sessions are ready. This condition is only relevant when the sample clock rates, sample clock timebase rates, sample counts, holdoffs, and/or any delays for the acquisitions are different. When the sample clock rates, sample clock timebase rates, and/or the sample counts are different in acquisition sessions sharing the reference trigger, you should also set the holdoff properties for the reference trigger master using the instrument driver. Pause Triggers *nitclk.configure_for_homogeneous_triggers()* configures generation sessions that support pause triggers to share them, if the pause triggers are None (no trigger configured) for all except one session. The pause triggers are shared by: - Implicitly exporting the pause trigger from the session whose script trigger is not None - Configuring the other sessions that support the pause trigger for digital-edge pause triggers with sources corresponding to the exported pause trigger - Setting *nitclk.SessionReference.pause_trigger_master_session* to the session that is exporting the trigger for all sessions that support script triggers If the pause triggers are configured for all generation sessions that support pause triggers, *nitclk.configure_for_homogeneous_triggers()* does not affect pause triggers. Pause triggers are considered to be configured for all sessions if either one or the other of the following conditions is true: - No session has a pause trigger that is None - One session has a pause trigger that is None and all other sessions have pause triggers other than None. The one session with the None trigger must have *nitclk.SessionReference.pause_trigger_master_session* set to itself, indicating that the session itself is the pause trigger master Note: TClk synchronization is not supported for pause triggers on acquisition sessions.

Parameters

sessions (*list of instrument-specific sessions or nitclk.SessionReference instances*) – sessions is an array of sessions that are being synchronized.

finish_sync_pulse_sender_synchronize

nitclk.**finish_sync_pulse_sender_synchronize**(*sessions*, *min_time=hightime.timedelta(seconds=0.0)*)

Finishes synchronizing the Sync Pulse Sender.

Parameters

- **sessions** (*list of instrument-specific sessions or nitclk.SessionReference instances*) – sessions is an array of sessions that are being synchronized.
- **min_time** (*hightime.timedelta, datetime.timedelta, or float in seconds*) – Minimal period of TClk, expressed in seconds. Supported values are between 0.0 s and 0.050 s (50 ms). Minimal period for a single chassis/PC is 200 ns. If the specified value is less than 200 ns, NI-TClk automatically coerces minTime to 200 ns. For multichassis synchronization, adjust this value to account for propagation delays through the various devices and cables.

initiate

nitclk.**initiate**(*sessions*)

Initiates the acquisition or generation sessions specified, taking into consideration any special requirements needed for synchronization. For example, the session exporting the TClk-synchronized start trigger is not initiated until after *nitclk.initiate()* initiates all the sessions that import the TClk-synchronized start trigger.

Parameters

sessions (*list of instrument-specific sessions or nitclk.SessionReference instances*) – sessions is an array of sessions that are being synchronized.

is_done

nitclk.**is_done**(*sessions*)

Monitors the progress of the acquisitions and/or generations corresponding to sessions.

Parameters

sessions (*list of instrument-specific sessions or nitclk.SessionReference instances*) – sessions is an array of sessions that are being synchronized.

Return type

bool

Returns

Indicates that the operation is done. The operation is done when each session has completed without any errors or when any one of the sessions reports an error.

setup_for_sync_pulse_sender_synchronize

nitclk.**setup_for_sync_pulse_sender_synchronize**(*sessions*, *min_time=hightime.timedelta(seconds=0.0)*)

Configures the TClks on all the devices and prepares the Sync Pulse Sender for synchronization

Parameters

- **sessions** (*list of instrument-specific sessions or nitclk.SessionReference instances*) – sessions is an array of sessions that are being synchronized.
- **min_time** (*hightime.timedelta, datetime.timedelta, or float in seconds*) – Minimal period of TClk, expressed in seconds. Supported values are between 0.0 s and 0.050 s (50 ms). Minimal period for a single chassis/PC is 200 ns. If the specified value is less than 200 ns, NI-TClk automatically coerces minTime to 200 ns. For multichassis synchronization, adjust this value to account for propagation delays through the various devices and cables.

synchronize

nitclk.**synchronize**(*sessions*, *min_tclk_period=hightime.timedelta(seconds=0.0)*)

Synchronizes the TClk signals on the given sessions. After *nitclk.synchronize()* executes, TClk signals from all sessions are synchronized. Note: Before using this NI-TClk method, verify that your system is configured as specified in the PXI Trigger Lines and RTSI Lines topic of the NI-TClk Synchronization Help. You can locate this help file at Start>>Programs>>National Instruments>>NI-TClk.

Parameters

- **sessions** (*list of instrument-specific sessions or nitclk.SessionReference instances*) – sessions is an array of sessions that are being synchronized.
- **min_tclk_period** (*hightime.timedelta, datetime.timedelta, or float in seconds*) – Minimal period of TClk, expressed in seconds. Supported values are between 0.0 s and 0.050 s (50 ms). Minimal period for a single chassis/PC is 200 ns. If the specified value is less than 200 ns, NI-TClk automatically coerces minTime to 200 ns. For multichassis synchronization, adjust this value to account for propagation delays through the various devices and cables.

synchronize_to_sync_pulse_sender

nitclk.**synchronize_to_sync_pulse_sender**(*sessions*, *min_time=hightime.timedelta(seconds=0.0)*)

Synchronizes the other devices to the Sync Pulse Sender.

Parameters

- **sessions** (*list of instrument-specific sessions or nitclk.SessionReference instances*) – sessions is an array of sessions that are being synchronized.
- **min_time** (*hightime.timedelta, datetime.timedelta, or float in seconds*) – Minimal period of TClk, expressed in seconds. Supported values are between 0.0 s and 0.050 s (50 ms). Minimal period for a single chassis/PC is 200 ns. If the specified value is less than 200 ns, NI-TClk automatically coerces minTime to 200 ns. For multichassis synchronization, adjust this value to account for propagation delays through the various devices and cables.

wait_until_done

nitclk.**wait_until_done**(*sessions*, *timeout=hightime.timedelta(seconds=0.0)*)

Call this method to pause execution of your program until the acquisitions and/or generations corresponding to sessions are done or until the method returns a timeout error. *nitclk.wait_until_done()* is a blocking method that periodically checks the operation status. It returns control to the calling program if the operation completes successfully or an error occurs (including a timeout error). This method is most useful for finite data operations that you expect to complete within a certain time.

Parameters

- **sessions** (*list of instrument-specific sessions or nitclk.SessionReference instances*) – sessions is an array of sessions that are being synchronized.
- **timeout** (*hightime.timedelta, datetime.timedelta, or float in seconds*) – The amount of time in seconds that *nitclk.wait_until_done()* waits for the sessions to complete. If timeout is exceeded, *nitclk.wait_until_done()* returns an error.

SessionReference

class nitclk.**SessionReference**(*session_number*)

Helper class that contains all NI-TClk properties. This class is what is returned by any nimi-python Session class tclk attribute when the driver supports NI-TClk

```
with niscope.Session('dev1') as session:
    session.tclk.sample_clock_delay = .42
```

..note:: Constructing this class is an advanced use case and should not be needed in most circumstances.

Parameters
session_number (*int, nimi-python Session class, SessionReference*) – nitclk session

exported_sync_pulse_output_terminal

nitclk.SessionReference.**exported_sync_pulse_output_terminal**

Specifies the destination of the Sync Pulse. This property is most often used when synchronizing a multichassis system. Values Empty string. Empty string is a valid value, indicating that the signal is not exported. PXI Devices - 'PXI_Trig0' through 'PXI_Trig7' and device-specific settings PCI Devices - 'RTSI_0' through 'RTSI_7' and device-specific settings Examples of Device-Specific Settings - NI PXI-5122 supports 'PFI0' and 'PFI1' - NI PXI-5421 supports 'PFI0', 'PFI1', 'PFI4', and 'PFI5' - NI PXI-6551/6552 supports 'PFI0', 'PFI1', 'PFI2', and 'PFI3' Default Value is empty string

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Export Sync Pulse Output Terminal**
- C Attribute: **NITCLK_ATTR_EXPORTED_SYNC_PULSE_OUTPUT_TERMINAL**

exported_tclk_output_terminal

nitclk.SessionReference.**exported_tclk_output_terminal**

Specifies the destination of the device's TClk signal. Values Empty string. Empty string is a valid value, indicating that the signal is not exported. PXI Devices - 'PXI_Trig0' through 'PXI_Trig7' and device-specific settings PCI Devices - 'RTSI_0' through 'RTSI_7' and device-specific settings Examples of Device-Specific Settings - NI PXI-5122 supports 'PFI0' and 'PFI1' - NI PXI-5421 supports 'PFI0', 'PFI1', 'PFI4', and 'PFI5' - NI PXI-6551/6552 supports 'PFI0', 'PFI1', 'PFI2', and 'PFI3' Default Value is empty string

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Output Terminal**
- C Attribute: **NITCLK_ATTR_EXPORTED_TCLK_OUTPUT_TERMINAL**

pause_trigger_master_session

nitclk.SessionReference.**pause_trigger_master_session**

Specifies the pause trigger master session. For external triggers, the session that originally receives the trigger. For None (no trigger configured) or software triggers, the session that originally generates the trigger.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	instrument-specific session or an instance of nitclk.SessionReference
Permissions	read-write

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Pause Trigger Master Session**
- C Attribute: **NITCLK_ATTR_PAUSE_TRIGGER_MASTER_SESSION**

ref_trigger_master_session

nitclk.SessionReference.**ref_trigger_master_session**

Specifies the reference trigger master session. For external triggers, the session that originally receives the trigger. For None (no trigger configured) or software triggers, the session that originally generates the trigger.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	instrument-specific session or an instance of nitclk.SessionReference
Permissions	read-write

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Reference Trigger Master Session**
- C Attribute: **NITCLK_ATTR_REF_TRIGGER_MASTER_SESSION**

sample_clock_delay

nitclk.SessionReference.**sample_clock_delay**

Specifies the sample clock delay. Specifies the delay, in seconds, to apply to the session sample clock relative to the other synchronized sessions. During synchronization, NI-TClk aligns the sample clocks on the synchronized devices. If you want to delay the sample clocks, set this property before calling *nitclk.synchronize()*. not supported for acquisition sessions. Values - Between minus one and plus one period of the sample clock. One sample clock period is equal to (1/sample clock rate). For example, for a session with sample rate of 100 MS/s, you can specify sample clock delays between -10.0 ns and +10.0 ns. Default Value is 0

Note: Sample clock delay is supported for generation sessions only; it is

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	hightime.timedelta, datetime.timedelta, or float in seconds
Permissions	read-write

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Sample Clock Delay**
- C Attribute: **NITCLK_ATTR_SAMPLE_CLOCK_DELAY**

sequencer_flag_master_session

nitclk.SessionReference.**sequencer_flag_master_session**

Specifies the sequencer flag master session. For external triggers, the session that originally receives the trigger. For None (no trigger configured) or software triggers, the session that originally generates the trigger.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	instrument-specific session or an instance of nitclk.SessionReference
Permissions	read-write

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Sequencer Flag Master Session**
- C Attribute: **NITCLK_ATTR_SEQUENCER_FLAG_MASTER_SESSION**

start_trigger_master_session

nitclk.SessionReference.**start_trigger_master_session**

Specifies the start trigger master session. For external triggers, the session that originally receives the trigger. For None (no trigger configured) or software triggers, the session that originally generates the trigger.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	instrument-specific session or an instance of nitclk.SessionReference
Permissions	read-write

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Start Trigger Master Session**
- C Attribute: **NITCLK_ATTR_START_TRIGGER_MASTER_SESSION**

sync_pulse_clock_source

nitclk.SessionReference.**sync_pulse_clock_source**

Specifies the Sync Pulse Clock source. This property is typically used to synchronize PCI devices when you want to control RTSI 7 yourself. Make sure that a 10 MHz clock is driven onto RTSI 7. Values PCI Devices - 'RTSI_7' and 'None' PXI Devices - 'PXI_CLK10' and 'None' Default Value - 'None' directs *nitclk.synchronize()* to create the necessary routes. For PCI, one of the synchronized devices drives a 10 MHz clock on RTSI 7 unless that line is already being driven.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Sync Pulse Clock Source**
- C Attribute: **NITCLK_ATTR_SYNC_PULSE_CLOCK_SOURCE**

sync_pulse_sender_sync_pulse_source

nitclk.SessionReference.**sync_pulse_sender_sync_pulse_source**

Specifies the external sync pulse source for the Sync Pulse Sender. You can use this source to synchronize the Sync Pulse Sender with an external non-TClk source. Values Empty string. Empty string is a valid value, indicating that the signal is not exported. PXI Devices - 'PXI_Trig0' through 'PXI_Trig7' and device-specific settings PCI Devices - 'RTSI_0' through 'RTSI_7' and device-specific settings Examples of Device-Specific Settings - NI PXI-5122 supports 'PFI0' and 'PFI1' - NI PXI-5421 supports 'PFI0', 'PFI1', 'PFI4', and 'PFI5' - NI PXI-6551/6552 supports 'PFI0', 'PFI1', 'PFI2', and 'PFI3' Default Value is empty string

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **External Pulse Source**
- C Attribute: **NITCLK_ATTR_SYNC_PULSE_SENDER_SYNC_PULSE_SOURCE**

sync_pulse_source

nitclk.SessionReference.**sync_pulse_source**

Specifies the Sync Pulse source. This property is most often used when synchronizing a multichassis system. Values Empty string PXI Devices - 'PXI_Trig0' through 'PXI_Trig7' and device-specific settings PCI Devices - 'RTSI_0' through 'RTSI_7' and device-specific settings Examples of Device-Specific Settings - NI PXI-5122 supports 'PFI0' and 'PFI1' - NI PXI-5421 supports 'PFI0', 'PFI1', 'PFI2', and 'PFI3' - NI PXI-6551/6552 supports 'PFI0', 'PFI1', 'PFI2', and 'PFI3' Default Value - Empty string. This default value directs *nitclk.synchronize()* to set this property when all the synchronized devices are in one PXI chassis. To synchronize a multichassis system, you must set this property before calling *nitclk.synchronize()*.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	str
Permissions	read-write

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Sync Pulse Source**
- C Attribute: **NITCLK_ATTR_SYNC_PULSE_SOURCE**

tclk_actual_period

`nitclk.SessionReference.`**`tclk_actual_period`**

Indicates the computed TClk period that will be used during the acquisition.

The following table lists the characteristics of this property.

Characteristic	Value
Datatype	float
Permissions	read only

Tip: This property corresponds to the following LabVIEW Property or C Attribute:

- LabVIEW Property: **Period**
- C Attribute: **NITCLK_ATTR_TCLK_ACTUAL_PERIOD**

- *pause_trigger_master_session*
- *ref_trigger_master_session*
- *sample_clock_delay*
- *sequencer_flag_master_session*
- *start_trigger_master_session*
- *sync_pulse_clock_source*
- *sync_pulse_sender_sync_pulse_source*
- *sync_pulse_source*
- *tclk_actual_period*

Exceptions and Warnings

Error

exception nitclk.errors.**Error**

Base exception type that all NI-TClk exceptions derive from

DriverError

exception nitclk.errors.**DriverError**

An error originating from the NI-TClk driver

UnsupportedConfigurationError

exception nitclk.errors.**UnsupportedConfigurationError**

An error due to using this module in an usupported platform.

DriverNotInstalledError

exception nitclk.errors.**DriverNotInstalledError**

An error due to using this module without the driver runtime installed.

DriverTooOldError

exception nitclk.errors.**DriverTooOldError**

An error due to using this module with an older version of the NI-TClk driver runtime.

DriverTooNewError

exception nitclk.errors.**DriverTooNewError**

An error due to the NI-TClk driver runtime being too new for this module.

DriverWarning

exception nitclk.errors.**DriverWarning**

A warning originating from the NI-TClk driver

Examples

You can download all nitclk examples for latest version here

nitclk_niscope_synchronize_with_trigger.py

Listing 1: (nitclk_niscope_synchronize_with_trigger.py)

```
import argparse
import niscope
import nitclk
import sys
import time


def example(resource_name1, resource_name2, options):
    with niscope.Session(resource_name=resource_name1, options=options) as session1,␣
↪niscope.Session(resource_name=resource_name2, options=options) as session2:
        session_list = [session1, session2]
        for session in session_list:
            session.configure_vertical(range=1.0, coupling=niscope.VerticalCoupling.DC)
            session.configure_horizontal_timing(min_sample_rate=50000000, min_num_
↪pts=1000, ref_position=50.0, num_records=1, enforce_realtime=True)
        session1.trigger_type = niscope.TriggerType.SOFTWARE
        nitclk.configure_for_homogeneous_triggers(session_list)
        nitclk.synchronize(session_list, 200e-9)
        nitclk.initiate(session_list)
        time.sleep(100)
        session1.send_software_trigger_edge(niscope.WhichTrigger.START)
        waveforms = session2.channels[0].fetch(num_samples=1000)
        for i in range(len(waveforms)):
            print(f'Waveform {i} information:')
            print(str(waveforms[i]) + '\n\n')


def _main(argsv):
    parser = argparse.ArgumentParser(description='Synchronizes multiple instruments to␣
↪one trigger.', formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument('-n1', '--resource-name1', default='PXI1Slot2', help='Resource␣
↪name of a NI Digitizer')
```

(continues on next page)

(continued from previous page)

```
    parser.add_argument('-n2', '--resource-name2', default='PXI1Slot3', help='Resource name of a NI Digitizer')
    parser.add_argument('-op', '--option-string', default='', type=str, help='Option string')
    args = parser.parse_args(argsv)
    example(args.resource_name1, args.resource_name2, args.option_string)


def main():
    _main(sys.argv[1:])


def test_example():
    options = {'simulate': True, 'driver_setup': {'Model': '5164', 'BoardType': 'PXIe', }, }
    example('PXI1Slot2', 'PXI1Slot13', options)


def test_main():
    cmd_line = ['--option-string', 'Simulate=1, DriverSetup=Model:5164; BoardType:PXIe',]
    _main(cmd_line)


if __name__ == '__main__':
    main()

```

4.2 Additional Documentation

Refer to your driver documentation for device-specific information and detailed API documentation.

Refer to the nimi-python Read the Docs project for documentation of versions 1.4.4 of the module or earlier.

CHAPTER

FIVE

LICENSE

nimi-python is licensed under an MIT-style license (see LICENSE). Other incorporated projects may be licensed under different licenses. All licenses allow for non-commercial and commercial use.

gRPC Features

For driver APIs that support it, passing a GrpcSessionOptions instance as a parameter to Session.__init__() is subject to the NI General Purpose EULA (see NILICENSE).

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

n

INDEX