labscript

Release 3.0.0rc2.dev29+ga805b69

labscript suite contributors

Jan 04, 2021

DOCUMENTATION

labscript, a component of the labscript suite, is an API used to define the experiment logic of a buffered experiment shot. This documentation will outline the general device hierarchy used when defining a connection table, and the **labscript** classes used to command input and output. For device specific documentation, and documentation for adding support for new devices, please refer to the labscript-devices documentation.

CHAPTER

ONE

INTRODUCTION

The labscript API is used to define the logic of an experiment that you wish to run. It is recommended that you read our paper before this documentation, so you are familiar with terms like pseudoclock. It would also be a good idea to familiarise yourself with the Python programming language and object oriented (OO) programming if you are not already.

To give you an idea of what a sample experiment looks like, the simplest experiment script (that does something) using the labscript API is below:

```
from labscript import *
from labscript_devices.PulseBlaster import PulseBlaster

# Connection Table
PulseBlaster(name='pulseblaster_0', board_number=0)
DigitalOut(name='my_digital_out', parent_device=pulseblaster_0.direct_outputs,␣
↪connection='flag 2')

#Experiment Logic
start()
my_digital_out.go_low(t=0)   # start low at the start
my_digital_out.go_high(t=1)  # go high at 1s
stop(2)                      # stop at 2s
```

The script consists of two parts, the connection table and the experiment logic which will be discussed in the following sections.

CHAPTER

TWO

CONNECTION TABLE

The connection table maps out the way input/output devices are connected to each other in your lab, and the channels (individual inputs/outputs) they have. The devices in your lab should be connected in a similar way to that shown in the figure below.

TODO: insert figure!

Here we see two *PseudoclockDevice* instances in the top tier of the diagram. They do not have a parent device that tells them when to update their output (this is true for all *PseudoclockDevice* instances). However, all but one (the master pseudoclock device) must be triggered by an output clocked by the master pseudoclock device.

Each *PseudoclockDevice* instance should have one or more *Pseudoclock* children. Some *PseudoclockDevice* instances may automatically create these children for you (check the device specific documentation). In turn, each *Pseudoclock* will have one of more *ClockLine* instances connected to it. These *ClockLine* instances generally refer to physical outputs of a device which will be used to clock another device. However, in some cases, one or more *ClockLine* instances may be internally created for you (check the device specific documentation).

If a device is not a *PseudoclockDevice*, it must be connected to one via a clockline. such devices inherit from *IntermediateDevice*. Inputs and outputs are then connected to these devices. If a *PseudoclockDevice* also has outputs that are not used for a *ClockLine*, then an *IntermediateDevice* is internally instantiated, and should be made available through the `PseudoclockDevice.direct_outputs` attribute (for example see PulseBlaster implementation TODO: link!).

CHAPTER

THREE

API REFERENCE

3.1 Device

class labscript.**Device**(*name*, *parent_device*, *connection*, *call_parents_add_device=True*, *added_properties={}*, *gui=None*, *worker=None*, *start_order=None*, *stop_order=None*, ***kwargs*)

Bases: object

__init__(*name*, *parent_device*, *connection*, *call_parents_add_device=True*, *added_properties={}*, *gui=None*, *worker=None*, *start_order=None*, *stop_order=None*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

get_properties(*location=None*)

Get all properties in location

If location is None we return all keys

set_properties(*properties_dict*, *property_names*, *overwrite=False*)

Add one or a bunch of properties packed into properties_dict

property_names is a dictionary {key:val, ...} where each val is a list [var1, var2, ...] of variables to be pulled from properties_dict and added to the property with name key (it's location)

property t0

The earliest time output can be commanded from this device at the start of the experiment. This is nonzeo on secondary pseudoclock devices due to triggering delays.

3.2 PseudoclockDevice

class labscript.**PseudoclockDevice**(*name*, *trigger_device=None*, *trigger_connection=None*, ***kwargs*)

Bases: labscript.labscript.TriggerableDevice

__init__(*name*, *trigger_device=None*, *trigger_connection=None*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

do_checks(*outputs*)

Basic error checking to ensure the user's instructions make sense

get_properties(*location=None*)

Get all properties in location

If location is None we return all keys

set_properties(*properties_dict*, *property_names*, *overwrite=False*)
Add one or a bunch of properties packed into properties_dict

property_names is a dictionary {key:val, ...} where each val is a list [var1, var2, ...] of variables to be pulled from properties_dict and added to the property with name key (it's location)

property t0
The earliest time output can be commanded from this device at the start of the experiment. This is nonzeo on secondary pseudoclock devices due to triggering delays.

trigger(*t*, *duration*, *wait_delay=0*)
Ask the trigger device to produce a digital pulse of a given duration to trigger this pseudoclock

3.3 Pseudoclock

class labscript.**Pseudoclock**(*name*, *pseudoclock_device*, *connection*, ***kwargs*)
Bases: labscript.labscript.Device

__init__(*name*, *pseudoclock_device*, *connection*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

collect_change_times(*all_outputs*, *outputs_by_clockline*)
Asks all connected outputs for a list of times that they change state. Takes the union of all of these times. Note that at this point, a change from holding-a-constant-value to ramping-through-values is considered a single state change. The clocking times will be filled in later in the expand_change_times function, and the ramp values filled in with expand_timeseries.

expand_change_times(*all_change_times*, *change_times*, *outputs_by_clockline*)
For each time interval delimited by change_times, constructs an array of times at which the clock for this device needs to tick. If the interval has all outputs having constant values, then only the start time is stored. If one or more outputs are ramping, then the clock ticks at the maximum clock rate requested by any of the outputs. Also produces a higher level description of the clocking; self.clock. This list contains the information that facilitates programming a pseudo clock using loops.

get_properties(*location=None*)
Get all properties in location

If location is None we return all keys

set_properties(*properties_dict*, *property_names*, *overwrite=False*)
Add one or a bunch of properties packed into properties_dict

property_names is a dictionary {key:val, ...} where each val is a list [var1, var2, ...] of variables to be pulled from properties_dict and added to the property with name key (it's location)

property t0
The earliest time output can be commanded from this device at the start of the experiment. This is nonzeo on secondary pseudoclock devices due to triggering delays.

3.4 ClockLine

class labscript.**ClockLine**(*name*, *pseudoclock*, *connection*, *ramping_allowed=True*, ***kwargs*)
Bases: labscript.labscript.Device

__init__(*name*, *pseudoclock*, *connection*, *ramping_allowed=True*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

get_properties(*location=None*)
Get all properties in location

If location is None we return all keys

set_properties(*properties_dict*, *property_names*, *overwrite=False*)
Add one or a bunch of properties packed into properties_dict

property_names is a dictionary {key:val, ...} where each val is a list [var1, var2, ...] of variables to be pulled from properties_dict and added to the property with name key (it's location)

property t0
The earliest time output can be commanded from this device at the start of the experiment. This is nonzeo on secondary pseudoclock devices due to triggering delays.

3.5 IntermediateDevice

class labscript.**IntermediateDevice**(*name*, *parent_device*, ***kwargs*)
Bases: labscript.labscript.Device

__init__(*name*, *parent_device*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

get_properties(*location=None*)
Get all properties in location

If location is None we return all keys

set_properties(*properties_dict*, *property_names*, *overwrite=False*)
Add one or a bunch of properties packed into properties_dict

property_names is a dictionary {key:val, ...} where each val is a list [var1, var2, ...] of variables to be pulled from properties_dict and added to the property with name key (it's location)

property t0
The earliest time output can be commanded from this device at the start of the experiment. This is nonzeo on secondary pseudoclock devices due to triggering delays.

CHAPTER

FOUR

LABSCRIPT SUITE COMPONENTS

The *labscript suite* is modular by design, and is comprised of:

Table 1: Python libraries

	labscript — Expressive composition of hardware-timed experiments
	labscript-devices — Plugin architecture for controlling experiment hardware
	labscript-utils — Shared modules used by the *labscript suite*

Table 2: Graphical applications

	runmanager — Graphical and remote interface to parameterized experiments
	blacs — Graphical interface to scientific instruments and experiment supervision
	lyse — Online analysis of live experiment data
	runviewer — Visualize hardware-timed experiment instructions

INDEX

Symbols

C

D

E

G

I

P

S

T