labscript Documentation

Release 2.0.1

Monash University

Jun 16, 2020

Contents

`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 1

Contents

1.1 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.

1.2 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!).

1.3 API Reference

1.3.1 Device

1.3.2 PseudoclockDevice

1.3.3 Pseudoclock

1.3.4 ClockLine

1.3.5 IntermediateDevice

CHAPTER 2

Indices and tables

- genindex
- modindex
- search