PyCMDS documentation

Contents

1 Introduction

PyCMDS is a program written for Wright group data acquisition. Its design is inspired by Python, PyQT, and previous data acquisition softwares in the Wright group: COLORS (COntrol for LOts of Research in Spectroscopy) by Schuyler Kain and ps control by Kent Meyer.

PyCMDS is based around three major design principles. These are:

1. Object orientation
2. Threading
3. Emergent behavior

PyCMDS is meant to be a development platform of sorts as well. PyCMDS focuses on simplifying development for the three main 'pillars':

1. Hardware
2. DAQ
3. Modules

These will be discussed more in their own sections.

Threading Queues and Mutexes

2 Hardware

Each `driver` object needs to have a series of hooks for PyCMDS to grab. The following is a minimum viable `Driver` and `GUI` class for PyCMDS.

```
### import ####################################################################


import os
import imp

from PyQt4 import QtCore

import project.project_globals as g
main_dir = g.main_dir.read()
app = g.app.read()
import project.widgets as pw
import project.ini_handler as ini
ini = ini.opas
import project.classes as pc


### address ###################################################################


class OPA(pc.Address):

    def dummy(self):
        print 'hello world im a dummy method'


# list module path, module name, class name, initialization arguments, friendly name
hardware_dict = {'OPA1 micro': [os.path.join(main_dir, 'OPAs', 'pico', 'pico_opa.py'), 'pico_o
                 'OPA2 micro': [os.path.join(main_dir, 'OPAs', 'pico', 'pico_opa.py'), 'pico_o
                 'OPA3 micro': [os.path.join(main_dir, 'OPAs', 'pico', 'pico_opa.py'), 'pico_o
                 'OPA1 TOPAS-C': [os.path.join(main_dir, 'OPAs', 'TOPAS', 'TOPAS.py'), 'TOPAS'
                 'OPA2 TOPAS-C': [os.path.join(main_dir, 'OPAs', 'TOPAS', 'TOPAS.py'), 'TOPAS'
hardwares = []
for key in hardware_dict.keys():
    if ini.read('hardware', key):
        lis = hardware_dict[key]
        hardware_module = imp.load_source(lis[1], lis[0])
        hardware_class = getattr(hardware_module, lis[2])
        hardware_obj = pc.Hardware(hardware_class, lis[3], OPA, key, True, lis[4])
        hardwares.append(hardware_obj)

```

```
### gui ################################################################################

gui = pw.HardwareFrontPanel(hardwares, name='OPAs')
advanced_gui = pw.HardwareAdvancedPanel(hardwares, gui.advanced_button)
```

2.1 OPAs

2.1.1 OPA800

2.1.2 TOPAS-C

TOPAS API error codes

Code	Meaning
0	No error
1	Unknown error
2	No TOPAS USB devices found
3	Invalid device instance
4	Invalid device index
5	Buffer too small
6	Failed to get TOPAS USB serial number
7	Device already opened
8	Device failed to open
9	USB communication channel failed to open
10	USB read error
11	Motor configuration failed to load
12	Configuration file doesn't match board configuration
13	Transmission of parameters failed
14	Device with this serial number not found
15	Invalid interface card type
16	Device has not been opened
17	USB command failed to receive response
18	Wavelength cannot be set
19	Invalid motor index
20	Function is not supported by LPT card
21	Invalid stage code
22	Invalid stage code
23	Tuning curve file failed to load
24	Tuning curve file read error
25	Wrong tuning curve file version
26	Wrong tuning curve type
27	Invalid number of motors
28	Invalid number of interactions
29	OPA type mismatch
30	Invalid wavelength
31	Invalid grating motor index in OPA tuning file
32	Tuning curve type mismatch
33	Configuration file not found
34	Invalid wavelength for this combination

The TOPAS-C units have DRM which lock them in software to only support certain tuning interactions. This means that to do certain operations (sum frequency signal, fourth harmonic idler etc) we need to let the TOPAS drivers know that we are 'allowed' to do such interactions using special keys.

Interaction	OPA1 (10743)	OPA2 (10742)
NON-NON-NON-Sig	4A58510A3057613B	11AECC8AF79A7C50
NON-NON-NON-Idl	35DB155E07C92B9C	476BFB4584AC45A1
NON-NON-SH-Sig	44BAD020566A2CDE	355A20DB305F57B6
NON-SH-NON-Sig	D292D9BAF9C79B06	CBB2588B4C2C10D0
NON-NON-SH-Idl	57B7288BBB30893C	B94438228A019CCF
NON-NON-SF-Sig	0863BD8C1FDC7245	9D1C47004B454776
NON-NON-SF-Idl	475AE539463B819F	D7AF59906687F172
DF1-NON-NON-Sig	F74E90B116261ABC	B65E646BC288469E
NON-SH-SH-Sig	6A56D2DB9CB7E9F0	0C62EA9BFB6F417E
SH-SH-NON-Sig	53E789590809495B	D8A6B290CAC4F28C
NON-SH-SH-Idl	75249C749E7E937A	F6212FF83BE18730
SH-NON-SH-Idl	ACB4B8A91ACDC14D	C0D2D58D45C73EB6

2.2 Delays

In addition to the attributes and methods shared by all drivers, delay-type drivers must posses the following:

Methods:

1. `set_zero(zero)`

Attributes:

1. `zero_position` (`pc.Number`)

3 DAQ

NI PCI-6251

4 Modules

Thin gui..