CE System Cheat Sheet

CE Instrument is a collection of Utility objects.

Utility:	Contains all the methods for a single piece of hardware (ie, z_stage will contain all commands needed to move the motorized z stage)
Method:	Calling a method will perform some sort of function with the hardware (ie, read_z() will read the current z position of the motorized z stage)
Controllers:	Controllers are the base layer for the utilities. Multiple utilities may use the same controller. Each controller needs to be opened before starting.

CE System Methods				
method	input	return	description	usage
load_config	config file path	None	Reads the CE system configuration file and initiliazes all the controller and utility objects. This is NOT the micromanager configuration file.	ce_system.load_config(r'.\config\NikonTE300.cfg)
open_controllers	None	None	Opens the controller objects. This must be done prior to calling any utility function, but after load_config.	ce_system.open_controllers()
close_controllers	None	None	Close the controller objects. Do this at the end of the day or when you'd like to open the controllers in a different notebook or python kernel.	ce_system.close_controllers()
startup_utilities	None	None	Runs the startup method for each utility object in collection. Normally run following open_controllers.	ce_system.startup_utilities()
shutdown_utilities	None	None	Puts utility objects in their shutdown state. Normally run prior to running close_controllers.	ce_sytem.shutdown_utilities()
stop_ce	None	None	Command to stop all utilities related to Capillary Electrophoresis runs	ce_sytem.stop_ce()
outlet_pressure	L2/PressureControl.py		ce_system.outlet_pressure	
method	input	return	description	usage
rinse_pressure	None	None	Applies pressure to the chamber for capillary rinses	ce_system.outlet_pressure.rinse_pressure()
rinse_vacuum	None	None	Applies vacuum to the outlet chamber for capillary rinses	ce_system.outlet_pressure.rinse_vacuum()
release	None	None	Opens the outlet chamber to atmospheric pressure. Required for gravity injections.	ce_system.outlet_pressure.release()
seal	None	None	Seals the outlet chamber and will remain in its state prior to sealing (e.g., pressurized)	ce_system.outlet_pressure.seal()
startup	None	None	First seals the chamber closing all valves, then opens the release valve.	ce_system.startup()
shutdown	None	None	First seals the chamber closing all valves, then opens the release valve.	ce_system.shutdown()
stop	None	None	Seals the outlet chamber.	ce_system.stop()
get_status	None	string	Returns the current state of the pressure valves	state = ce_system.outlet_pressure.get_status()
xy_stage	L2/XYControl.py		ce_system.xy_stage	
method	input	return	description	usage
read_xy	None	[float, float]	Returns the position of the stage in mm	x,y = ce_system.xy_stage.read_xy()
set_xy	[float, float]	None	Sets the absolute position of the stage in mm	ce_system.xy_stage.set_xy([2,2]) # Moves the stage to 2mm , 2mm
set_rel_xy	[float, float]	None	Moves the stage by distance specified from its current position.	ce_system.xy_stage.set_rel_xy([0.1, 0]) # Moves the stage 0.1 mm in the positive x direction
set_home	None	NOne	Sets the current position as the home or (0,0) position.	ce_system.xy_stage.set_home()
go_home	None	None	Moves the stage to the Home or (0,0) position	
stop	None	None	Stops the stage in its current move. If a stop command is not available in the hardware, it will perform a relative move by 0,0 from its current position.	ce_system.xy_stage.stop()
objective, inlet_z, outlet_z	L2/ZControl.py		ce_system.objective, ce_system.inlet_z, ce_system.outlet_z	
Each of these utilities use the same utility module (Zcontrol). Their methods will be the same, you only need to access the correct object within the CESystem collection. Do this by substituting 'objective' for 'inlet_z' or				
method	Input	Return	Description	Usage

set_z	float	None	Move the stage to the absolute position specified by z in mm	ce_system.objective.set_z(10), ce_system.inlet_z.set_z(25), ce_system.outlet_z.set_z(250)
set_rel_z	float	None	Moves the stage by distance rel_z specified in mm	ce_system.objective.set_rel_z(1), ce_system.inlet_z.set_rel_z(-1), ce_system.outlet_z.set_rel_z(-10)
read_z	None	float	Reads the current position of the stage in mm	ce_system.objective.read_z(), ce_system.inlet_z.read_z(), ce_system.outlet_z.read_z()
set_home	None	None	Sets the current position as the home, or 0 position of the stage	ce_system.objective.set_home(), ce_system.inlet_z.set_home(), ce_system.outlet_z.set_home()
go_home	None	None	Go to the home or 0 positino of the stage (not a limit switch)	ce_system.objective.go_home(), ce_system.inlet_z.go_home(), ce_system.outlet_z.go_home()
homing	None	None	Home the stage using the limit switches if available. This command can only be called one time and may be done during the startup process if there is no risk to the capillary. The Thorlabs stage will not home during startup as its homing position may cause the capillary to run into the stage. This necessitates that the thorlabs stage homing be called manually.	ce_system.objective.homing(), ce_sytem.inlet_z.homing(), ce_system.outlet_z.homing()
stop	None	None	Stops the stage from moving. If the hardware does not have this command will perform a relative move by 0 distance from the current position.	ce_system.objective.stop(), ce_system.inlet_z.stop(), ce_system.inlet_z.stop()
filter_wheel	L2/FilterWheelControl.py		ce_system.filter_wheel, ce_system.excitation_wheel	
			Lumencor is controlled using the filter wheel abstraction instead of light control.	
method	input	return	description	usage
set_channel	int or str	None	Move the Filter wheel t the selected channel specified either by an integer or a character. **Lumencor**: Channel is a string corresponding to channel of light to allow through. Multiple channels are allowed.	ce_system.filter_wheel.set_channel(1), ce_system.excitation_wheel.set_channel(['C','R'])
get_channel	None	int or string	Returns the current filter wheel setting as either an int or a string.	chnl = ce_system.filter_wheel.get_channel()
get_status	None	dictionary	Returns a dictionary containing keyword 'filter', and value containing the filter wheel channel.	state = ce_system.filter_wheel.get_status()
high_voltage	L2/HighVoltageControl.py		ce_system.high_voltage	
method	input	return	description	usage
set_voltage	float	None	Sets the voltage of power supply using V specified in kilovolts	ce_system.high_voltage.set_voltage(15) # Sets voltage to 15 kV
get_voltage	None	float	Retrieves the last voltage reading for the power supply.	volt = ce_system.high_voltage.get_voltage()
get_current	None	float	Retrieves the last current reading from the powersupply in microaps	ua = ce_system.highe_voltage.get_current()
get_data	None	dictionary	Retrieves a dictionary containing a time series list of voltage and current readings as values for the corresponding 'voltage' and 'current' dictionary keys.	data = ce_system.high_voltage.sget_status()
lysis_laser	L2/LaserControl		ce_system.lysis_laser	
method				
laser_standby	None	None	Puts the laser into a ready state, the laser will be ready to fire	ce_system.lysis_laser.laser_standby()
laser_stop	None	None	Removes the laser from a ready state, the laser will not fire	ce_system.lysis_laser.laser_stop()
laser_fire	None	Boolean	Fires the laser if the laser is in a ready (standby) state	ce_system.lysis_laser.laser_fire()
laser_check	None	Boolean	Returns True if the laser is in the standby or ready state.	ce_system.lysis_laser.laser_check()
inlet_rgb	L2/LightControl.py		ce_system.inlet_rgb	
			Lumencor is controlled using the filter wheel abstraction instead of light control.	
method	input	return	description	

turn_on_channel	string or int	string	Turns on the red, green, or blue LED. ('R', 'G', 'B')	ce_system.inlet_rgb.turn_on_channel('R')
turn_off_channel	string or int	string	Turns off the red, green, or blue LED. ('R', 'G', 'B')	ce_system.inlet_rgb.turn_off_channel('R')