How to run

The code provided is based on the ref-step code. This guide contains basic running instructions, and information about the code.

# Getting started

First open the application by double clicking Main.py.

The software has three windows, one for the graph displays, one for information tables (bottom left) and one for control purposes.

In the control window, under the “instruments” tab, the instruments need to be matched up with GPIB addresses. To display currently available addresses, click the ‘Refresh address’ button.

Next, a control template needs to be loaded, this is done form the top tab under “File>Open File” button. Templates have been created for the fluke and CH sources (TEMPLATE-FLUKE and TEMPLACE-CH). The template loads a “Control settings” table, this has the actual values to send the instruments.

The graph can be used, but after a lot of data points are added it crashes the program. So perhaps it should only be used for testing out the setup, then the program can be restarted without running the graph.

Once data collecting is done, click “File>Save tables” and choose a name for the tables. The excel sheet saved will be identical to the template sheet, except with data saved into it too so that excel now does the calculations.

The code

The code is similar to that of ref-step, but uses specific instrument classes. The main files that a user might want to edit are the gpib\_data.py file, and all “inst\_” files. Gpib\_data contains the thread that communicates with all instruments and the wx table, so here the order of commands can be changed or additional commands can be added. For example, if the command to reset the lock in needs to be sent the user should add:

Self.com(self.lockin.reset)

The self.com function executes the command in the bracket (self.lockin.reset), and also receives info on whether or not the command was successful. If it was not successful, it will call the MakeSafe on the instruments and the program stops.

If a new command needs to be added to the lockin, the file “inst\_lockin.py” needs to be edited, a new function should be added as such:

def new\_function(self, optional\_argument):

return self.send(“The new words that need to be sent”+str(optional\_argument))

As the name suggests, the optional argument is optinoal. If a function with an argument is called using the “self.com” method (that checks the commands sent safely), the arguments need to be given to the com function too as follows: self.com(self.some\_isntrument.function, argumetns).

Note that sending an instrument the word ‘None’, will result in the instrument skipping the command but still printing to the event reports box that a command was skipped.

The instruments’ “send” function is defined in instrument.py, the instruments are all subclasses of that general class.

# Bugs

Like in ref-step, when large tables are loaded to the grid, the grid ends up creating its own scroll bar. When you mouse over that new scroll bar the computer crashes and the screen turns temporarily black. This can be avoided with smaller tables, but I don’t have a fix at the moment.