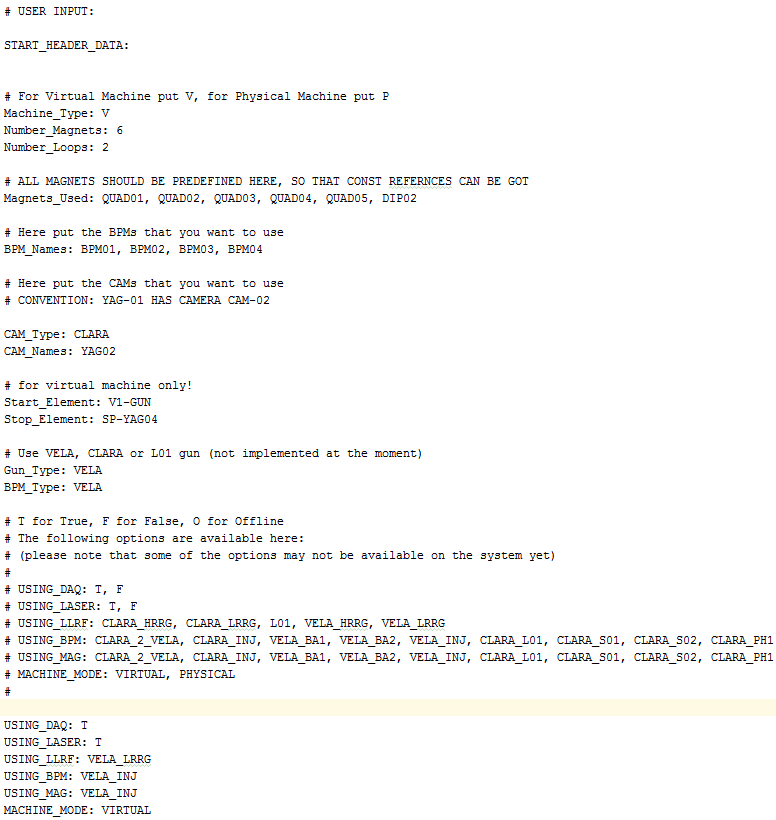
**Generic Experiment Python Script – Michael Sullivan**

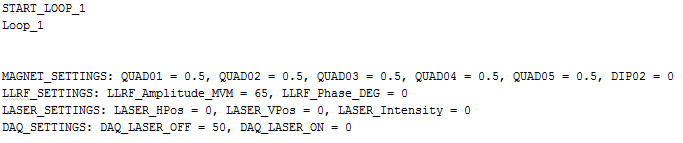
The Python project “Generic Experiment” is a structure of code that allows the user to automate their experiments using the VELA/CLARA lines via a predetermined and user-defined text document.

**The text file:**

The text document is set out into two parts: “header data” and “loop data”. The header data contains data such as the type of machine that you want to use (Physical or Virtual), the list of magnets, BPMs and cameras you want to use as well as what part of the machine you want to use them (e.g. CLARA\_INJ, VELA\_BA1, etc.). You can also set the LLRF and the Laser in this way. Also, as this script also works with the Virtual Machine (for the most part), the start and stop elements of the simulation must also be included. A template of the text file is included in the folder with the python script so it can be modified from experiment to experiment. A screenshot of the header data for a particular experiment is shown below, where “#” symbols at the start of a line denotes a comment which is not read in by the python script.



It can be seen in the screenshot that helpful hints are sometimes given for certain parts so that the user can understand that layout more easily. It is also possible that the user can add comments into their text document for their own reasons by putting a “#” at the beginning of the line.

The second part of the text document is the loop data, which is where all the information about the magnet, llrf, laser and data acquisition settings are stored. The idea of the script is that the machine will get the machine ready for performing experiments using the header data and then process through the loop settings one by one, setting the loop values in the machine and then taking data before moving onto the next loop. This can be easier seen by looking at an example of some loop settings below:

In this loop example of an experiment on the VELA line, the script will set the magnets with the values defined, e.g. magnet QUAD01 will be set with a current of 0.5 amps. Similarly, the llrf and laser will also be set up. The DAQ\_SETTINGS here tell the script how to acquire the data, for example the number assigned to DAQ\_LASER\_OFF is the number of shots that will be acquired of the beam (using the cameras defined in the header data) with the laser turned off. Similarly the number assigned to DAQ\_LASER\_ON will do the same thing but this time with the laser turned on. If the user does not want to run the experiment with the laser on, then they should set DAQ\_LASER\_ON = 0, i.e. there will be zero shots taken with the laser on.

The user can add as many loops onto the text file as they want, and after all loops have been completed, BPM data will be written to an output txt file with a timestamp and (raw) camera data will be acquired when using the physical machine.

**The script:**

The project consists of six python files:

* File\_reader:
  + Reads in the user-defined text file and stores all of the data into a dictionary which is then passed onto other parts of the script.
* Master\_controller:
  + This creates all of the necessary c++ controllers based on information in the text file which are then passed onto “setters”
* Parameter\_setter:
  + This is passed the magnet, laser and llrf controller objects and uses them to set the various values associated with them; currents, gradients, etc.
  + This setter also connects with references to “talk” with the machine to assure that values are set properly.
* Daq\_getter:
  + This is passed the bpm and camera daq controllers and uses them to read data from the machine and outputs to associated data files.
* Global\_keywords:
  + Essentially stores all the lists of keywords used to read the text file and other places, so if they need to be changed in the future they only need to be changed here as they are passed as variables in all other places.
* Main:
  + The main is the script that the user will use to actually run the whole program as it imports all the other files.
  + Here, the user should change the input file variable to match their own text file name and then everything should adapt to that txt file.
  + In this script, a master\_controller object is created which then acts as the “hub” and creates everything and passes controller objects to the necessary scripts.

A flow-style diagram is given below for easier viewing:

**Master\_controller**

**File\_reader**

**Parameter\_setter**

**Daq\_getter**

**Main**

Reads in text file

Passes magnet, llrf, laser controllers to set values

Passes BPM and camera controllers to gather data

Uses master controller object to run the program