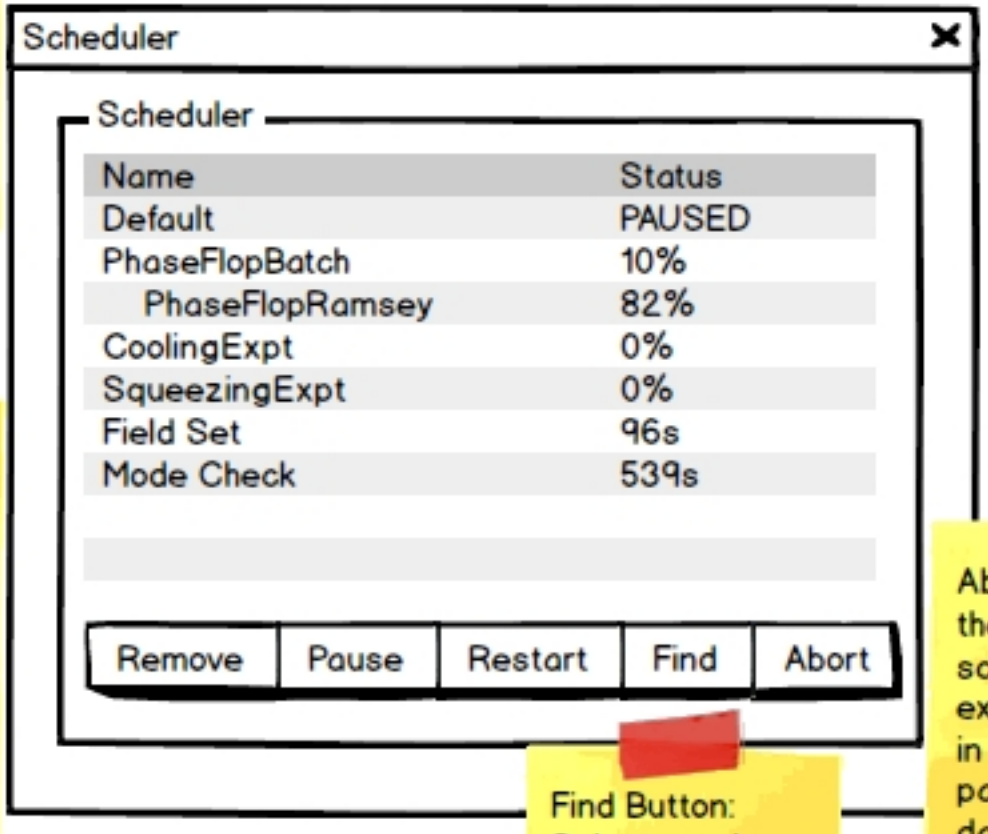
Some notes about scheduler functions:

The top of list shows what is currently executing - it is a FIFO stack except for the default experiment which always sits at the top and pauses for other experiements that execute. While experiments are waiting to execute in the stack (0%). To determine if an experiment is pauseable, a yeild command is specifically included in the experiment script.

There are other experiments that are not in the stack -- they execute every XX seconds. When they count down, they break into the current experiment if it is pauseable, else they jump to the top of the stack.



Find Button: Selects and displays Expt. Explorer window for a given scheduled experiement

Abort Button: clears the queue and schedules the experiement stored in the "abort" parameter in the database.

Experiment List

Debugging

Rand. Bench

ODF

Transport

+

- 1 Be
  - Expt One
  - Expt Two
  - Expt Three
- 2 Mg
  - MS Gate
  - Sz Gate
  - CZ Gate
  - I Gate
- Hardware
  - TTL
  - DDS
  - Waveplate
  - etc.

Show
Duplicate
Move
Delete

A menu that opens when RightClicking on an experiment:
Show: open Expt Explorer without parsing.
Duplicate: make a copy of experiment
Move: Move experiment to different tab
Delete: delete experiement

Selecting an experiment will generate the GUI associates with the experiment.

These experiment GUI (widgets) are highly configurable for each experiment: which widgets to be prompt and what are the functionality of those widgets.

These widget can be:

Sweep control
Plot control
Fit control

Run

Schedule

Parse

Automatically Parse

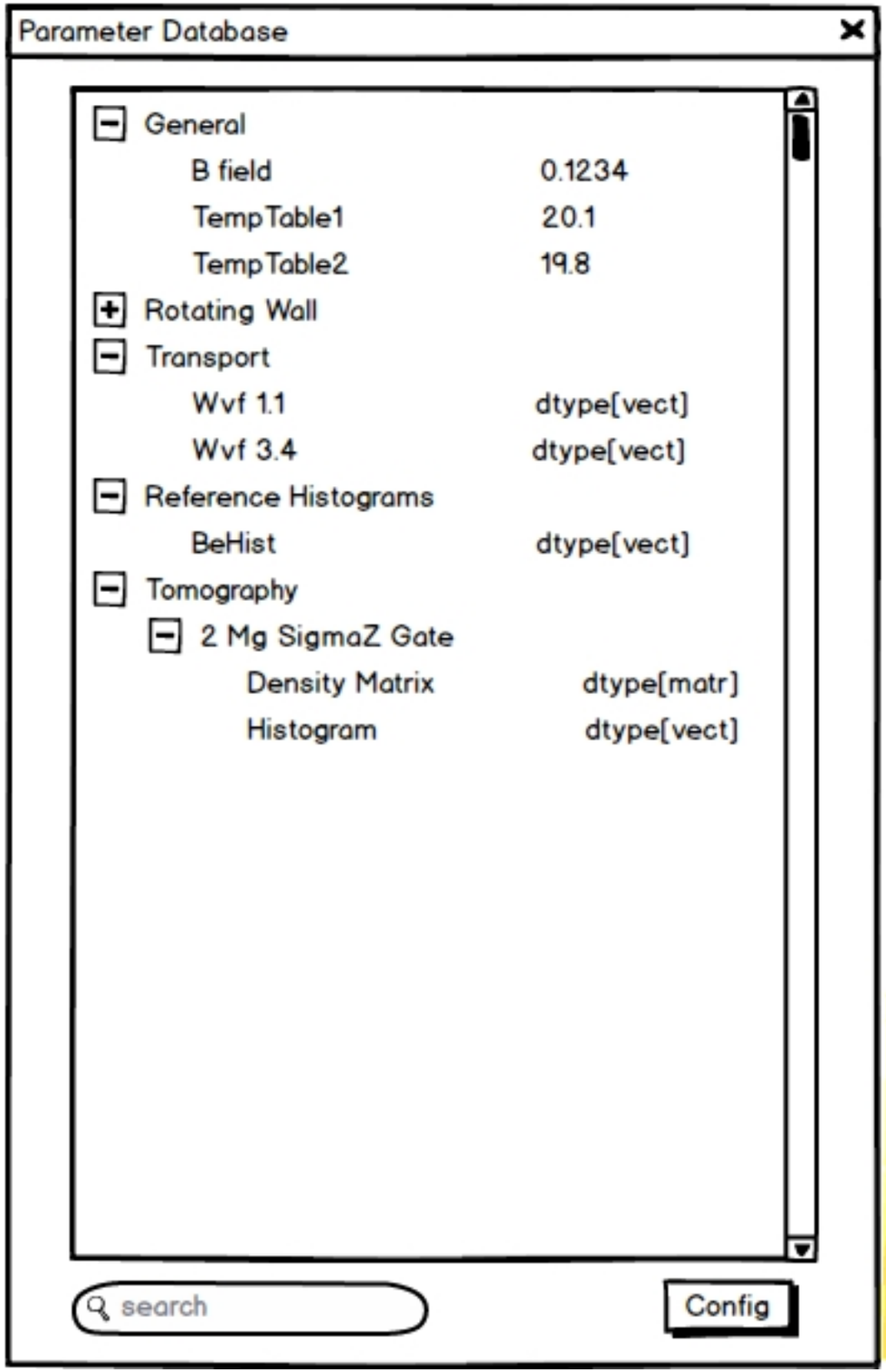
Run: Will immediately start running selected experiment if currently running is pauseable, else will go to top of the queue.

Schedule: Add to the bottom of the queue -- but will run before experiments that are waiting on a cycle time.

Parse: Will parse the experiment and show the Experiment Explorer window

Parameter database contains all the information needed to run experiments.

Entries can be defined as more than just scalars, such that the database can store reference histograms, PDQ waveforms, density matrices, matrices of parameters (e.g. the spin flip frequency each ion in a multi-ion experiment, or a matrix of the coupling paramters in a multi-ion experiment).

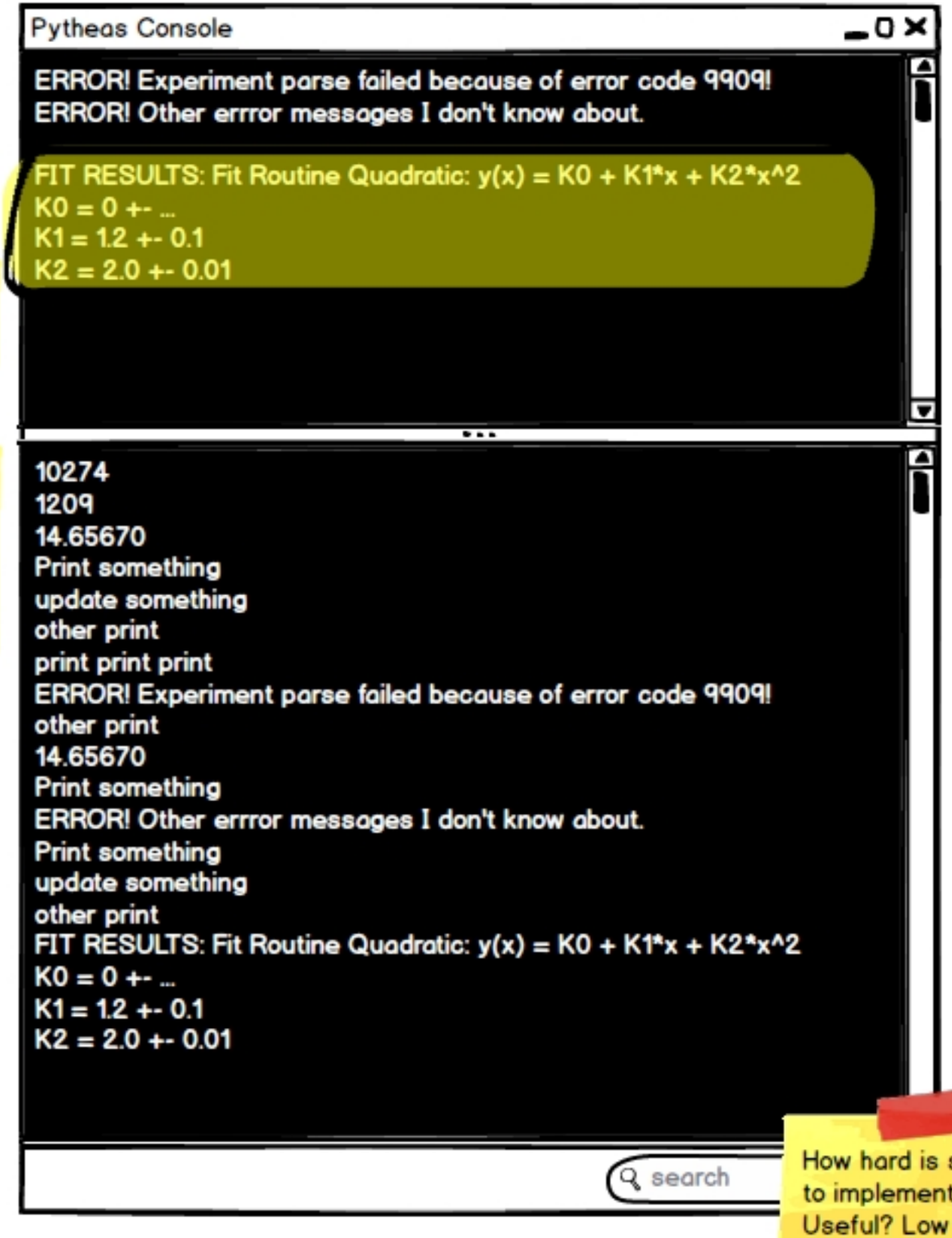


Config: Allow changing of the properties of the selected parameter:
name
class
numeric format: 0.01 or 1x10^-2
Console update configuration: on/off, frequency

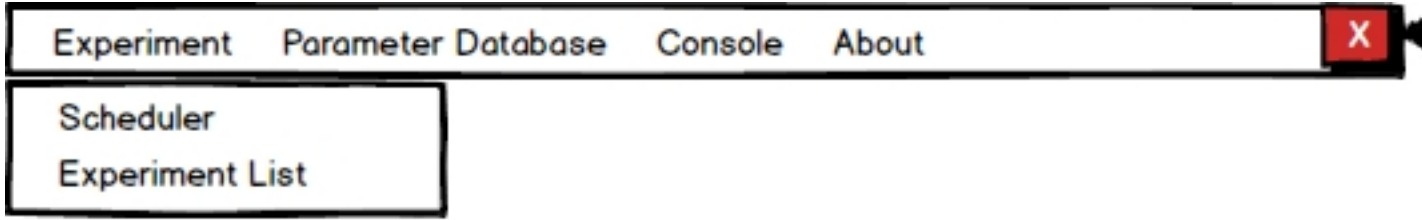
Prints to the console have priority, and the console can be figured to show windows of different level of priority. So in one window, only high priortty messages that you want to be able to be easy to read will show -- errors and fits for examples.

Can choose color of prints to the console

Other windows show full console information



How hard is search to implement? Useful? Low priority


Experiment
Parameter Database
Console
About
X
Scheduler
Experiment List
Prompt a window asking:
"Are you sure you want to quit?"
and gives you the option to save the layout.
** Other window X buttons will NOT close the GUI, just the window **

Your Fav Code Editor

```
class scan1Dexpt():
    parameters = "scanned_experiment scanned_parameter"

    def get_widgets():
        return [
            Checkbox("2nd Doppler Cooling", "enable_doppler_cool2"),
            GlorifiedSlider(self.scanned_parameter) ] +
self.scanned_experiment.get_widgets()

    def run(self):
        for x in range(start, stop)
            exp = self.scanned_experiment(self, **{self.scanned_parameter: x})
            yield exp
            broadcast(s, self.histogram)
            plot(xdata, ydata, o)
        if enableSave:
            nameAppend = getTime()
            self.saveData(ParamDB.saveFile+nameAppend)
```

GlorifiedSlider() is an example of function that returns a list of widgets packaged for purpose

SB has said that a class like scan1Dexp() would likely be something in a library provided by ARTIQ, so many of the details of how this would work would not need to be accessed day to day. Still, it shows the simplicity of adding widgets as your own classes can have a get_widgets() function.

Pytheas GUI Experiment Panel

myExperimentName

Experiment definition file

C:/qc/exptFiles/SidebandCooling.dc

☑ 2nd Doppler Cooling

☐ Save? Output: C:/qc/userData/output/

ExptToScan.QuickControls

☑ Boolean

1209 Text to change

Other Widgets determined by toolkit

These controls appear from the scanned_experiment.get_widgets() which is defined in the scanned_experiment class

Scan Widget

arm_t

final_p

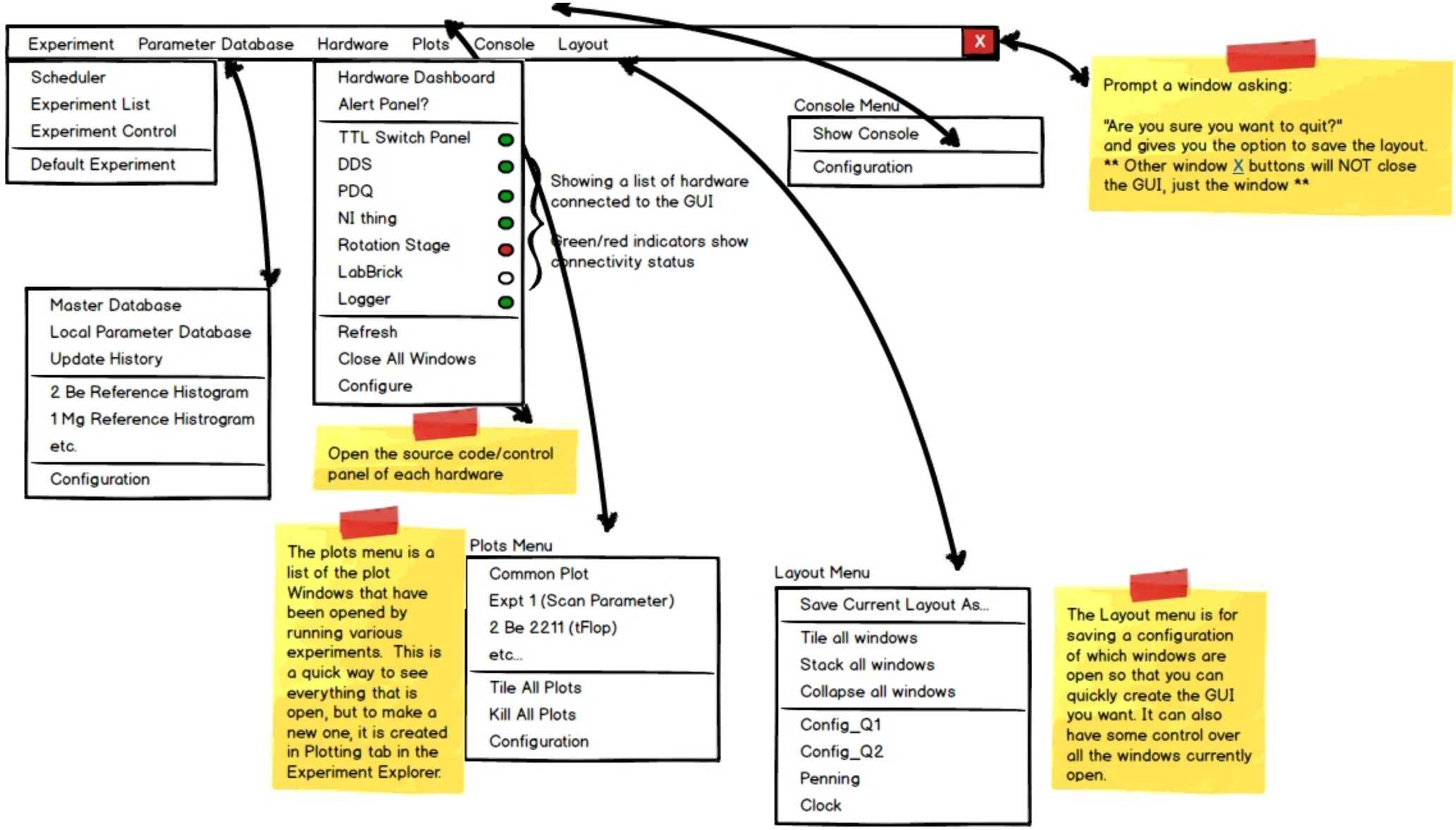
Start	Stop	Interval	Random?
0	15000	0.01	☐

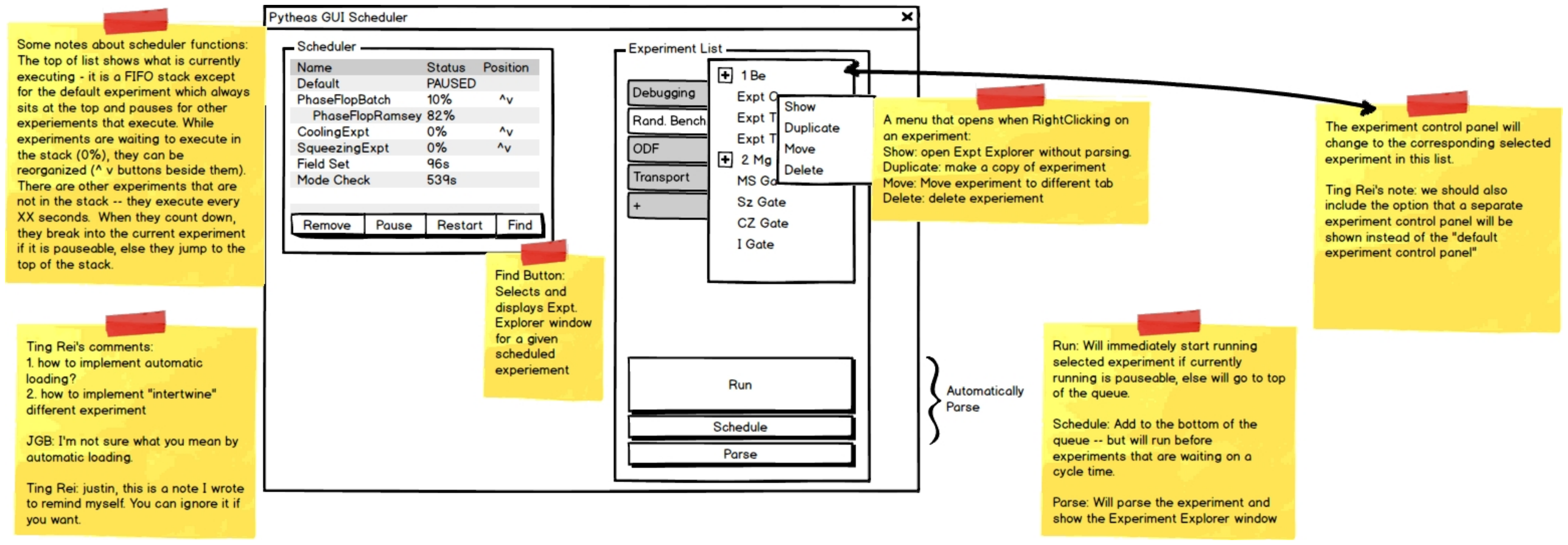
<--> CenterOn Cursor -><-

General Description

The master control panel of the GUI is a tab bar/menu bar with a list of items in different categories.

A separate window will open when click on these items.




Some notes about scheduler functions: The top of list shows what is currently executing - it is a FIFO stack except for the default experiment which always sits at the top and pauses for other experiements that execute. While experiments are waiting to execute in the stack (0%), they can be reorganized (^ v buttons beside them). There are other experiments that are not in the stack -- they execute every XX seconds. When they count down, they break into the current experiment if it is pauseable, else they jump to the top of the stack.
Pytheas GUI Scheduler
Scheduler
Name
Status
Position
Default
PAUSED
PhaseFlopBatch
10%
^v
PhaseFlopRamsey
82%
CoolingExpt
0%
^v
SqueezingExpt
0%
^v
Field Set
96s
Mode Check
539s
Remove
Pause
Restart
Find
Find Button: Selects and displays Expt. Explorer window for a given scheduled experiement
Experiment List
Debugging
Rand. Bench
ODF
Transport
+
1 Be
Expt C
Expt T
Expt T
2 Mg
MS Ga
Sz Gate
CZ Gate
I Gate
Show
Duplicate
Move
Delete
A menu that opens when RightClicking on an experiment:
Show: open Expt Explorer without parsing.
Duplicate: make a copy of experiment
Move: Move experiment to different tab
Delete: delete experiement
The experiment control panel will change to the corresponding selected experiment in this list.
Ting Rei's note: we should also include the option that a separate experiment control panel will be shown instead of the "default experiment control panel"
Run
Schedule
Parse
Automatically Parse
Run: Will immediately start running selected experiment if currently running is pauseable, else will go to top of the queue.
Schedule: Add to the bottom of the queue -- but will run before experiments that are waiting on a cycle time.
Parse: Will parse the experiment and show the Experiment Explorer window
Ting Rei's comments:
1. how to implement automatic loading?
2. how to implement "intertwine" different experiment
JGB: I'm not sure what you mean by automatic loading.
Ting Rei: justin, this is a note I wrote to remind myself. You can ignore it if you want.

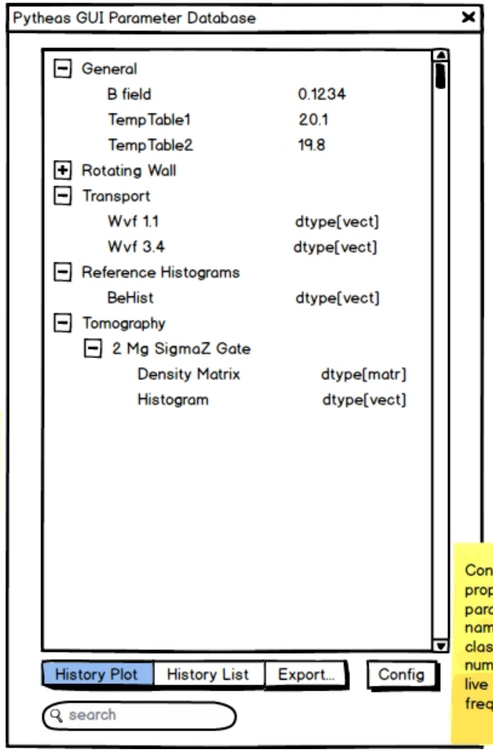
This first window is the main way of viewing and interacting with the paramter database. Entries can be defined as more than just scalars, such that the database can store reference histograms, PDQ waveforms, density matrices, matrices of parameters (e.g. the spin flip frequency each ion in a multi-ion experiment, or a matrix of the coupling paramters in a multi-ion experiment).

Reference histogram is saved every xxx minutes. User can access and look at how the each reference histogram behave.

History Plot: A simple plot of the selected paramter vs time. Can be used as a "logging" type plot. Of course, this database is on Git, so I suppose it can be accessed and plotted through other means (grafana? others?) but this could provide a simple interface built in. -- How to handle other datatypes?

History List: Display a .CSV of last 10 (20? 50?) data points for a parameter

Export: A quick way to extract data from the database and save it to another file

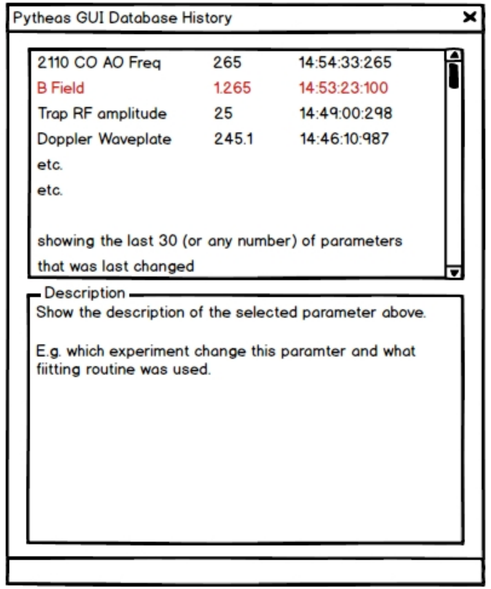


Config: Allow changing of the properties of the selected parameter:
name
class
numeric format: 0.01 or 1x10^-2
live feed configuration: on/off, frequency

These are 2 other windows connected to the database, but the idea is that they are "Display Only" to reduce complexity associated with interacting with the database.

A live feed of parameters that have been updated, regardless of which experiment or which protocol that update them.

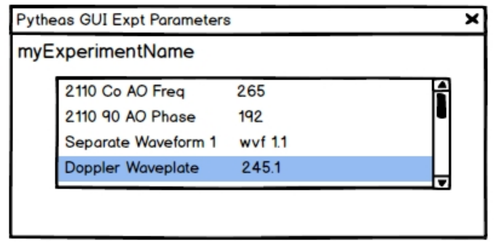
I removed the buttons here with the idea that this is just a 'display' -- to interact with a paramter you have to go to the database window.

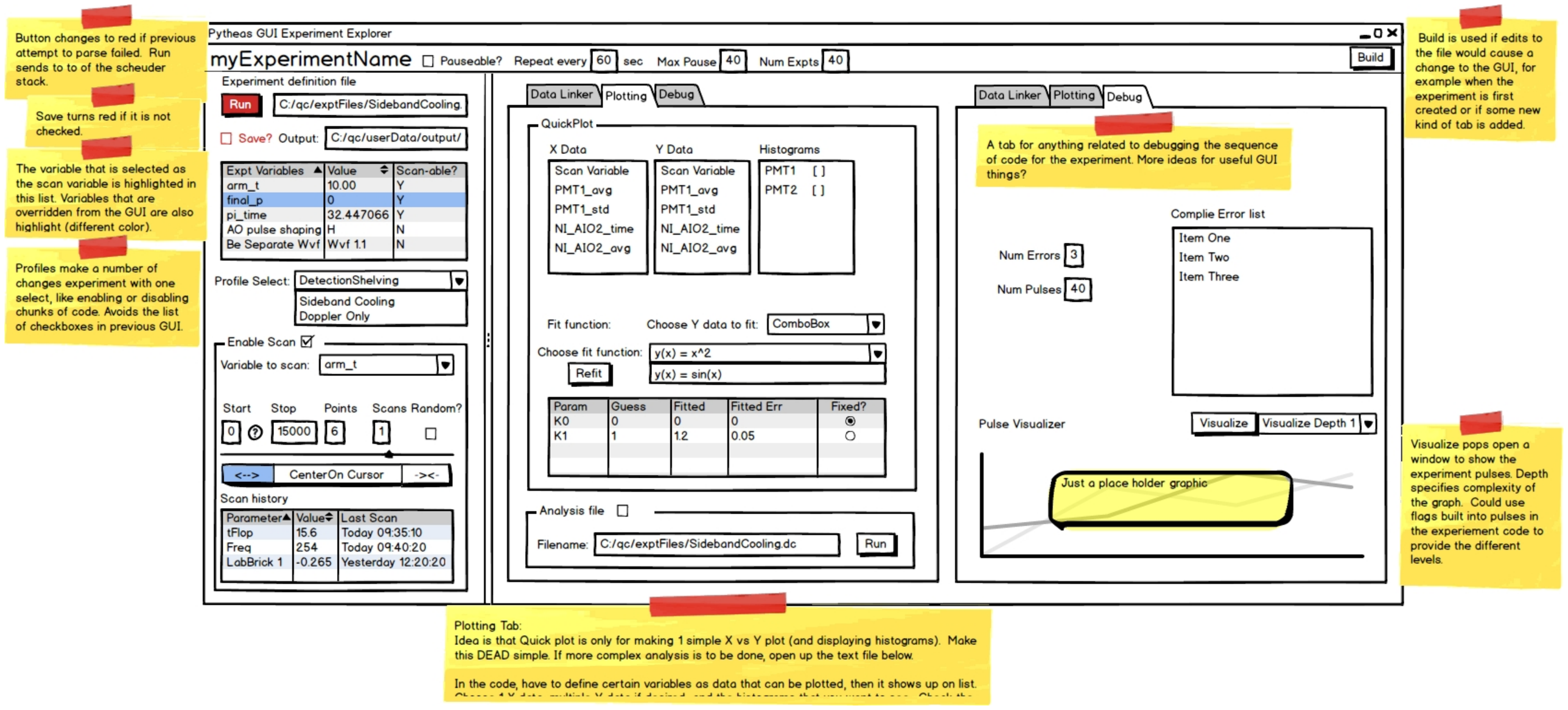


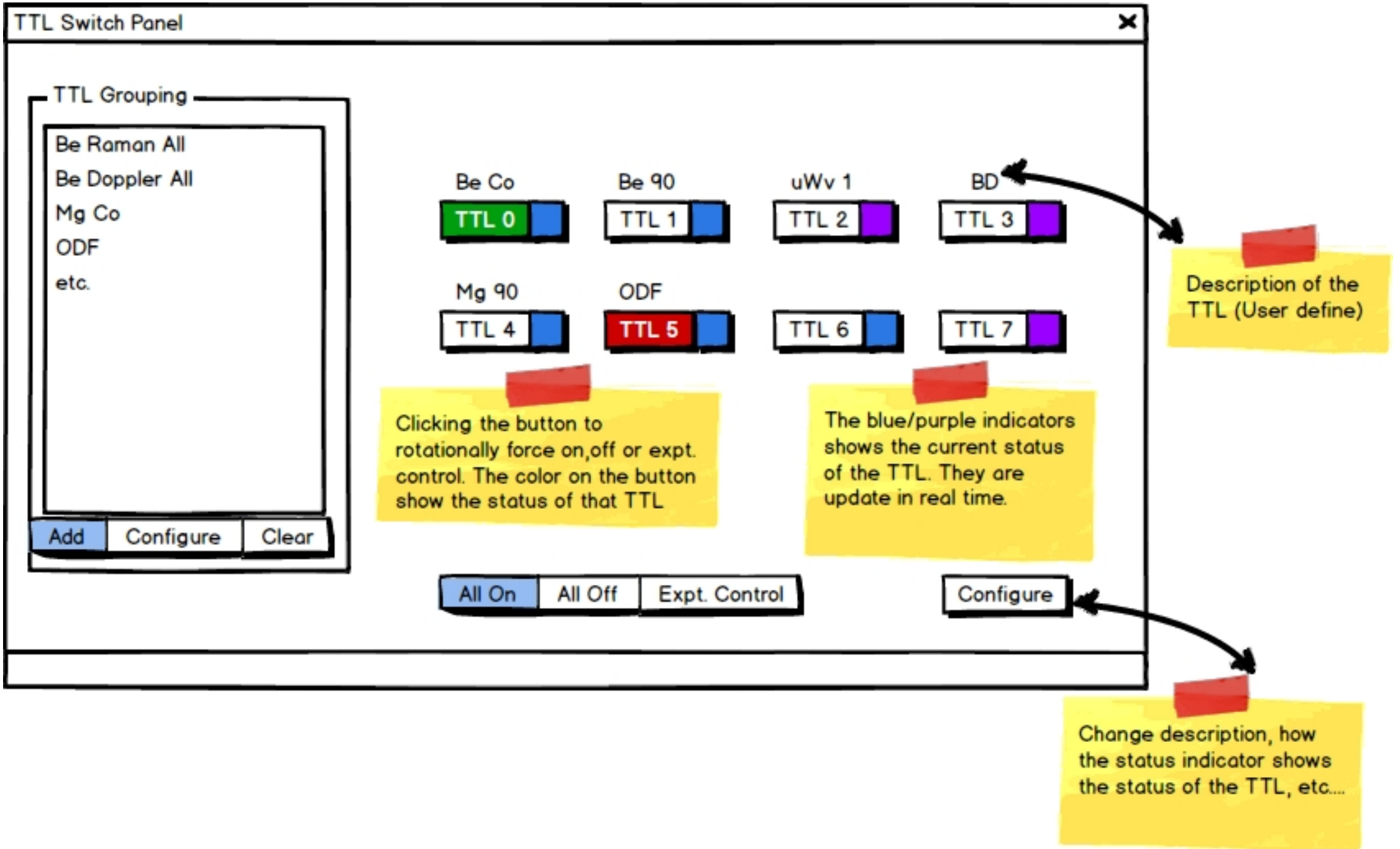
Showing the parameters used in the currently selected experiment (name displayed at the top).

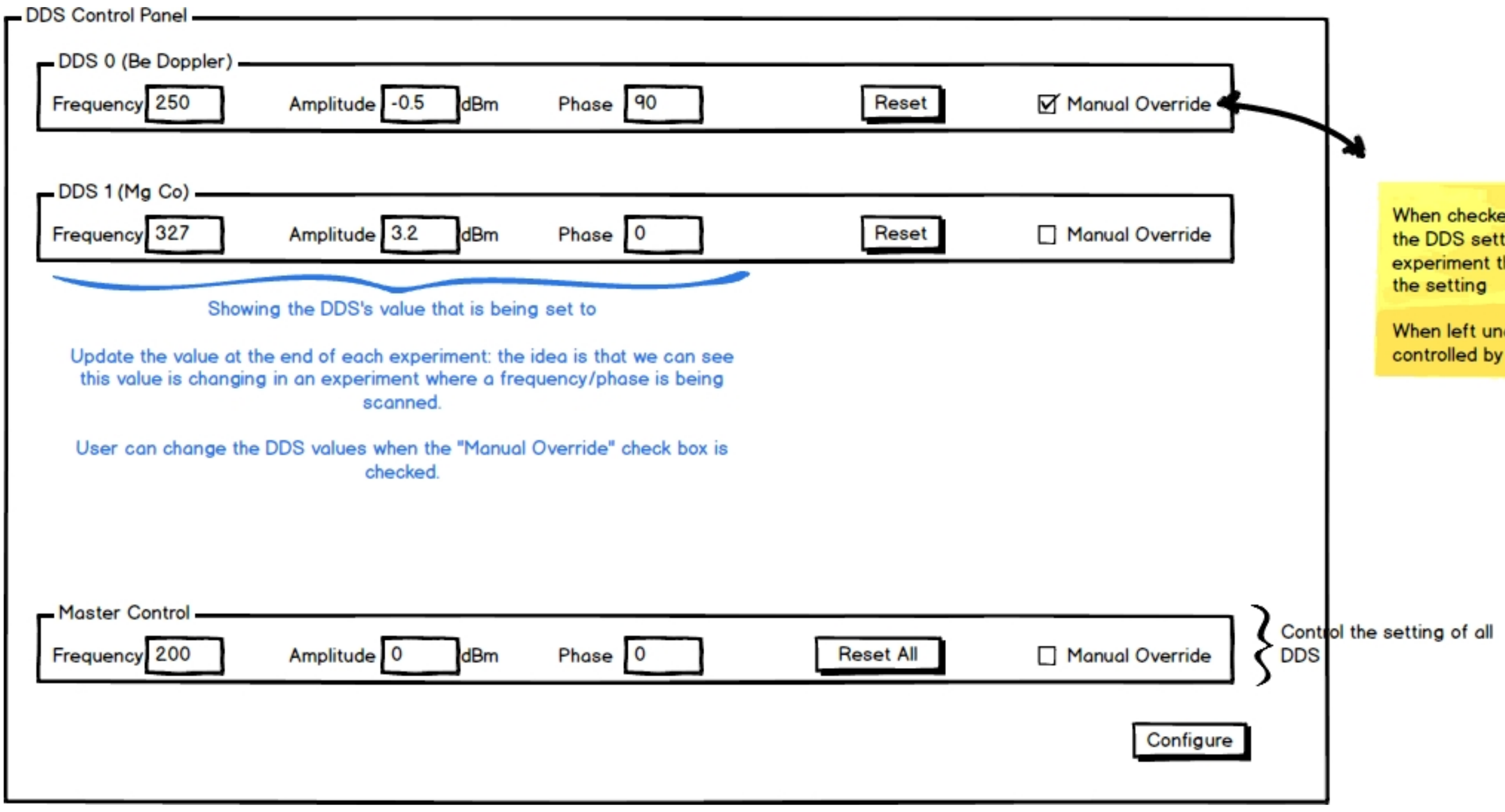
This is a subset of parameters of those in the master database.

highlighting for variables that will be written to by the selected experiment




Button changes to red if previous attempt to parse failed. Run sends to to of the scheuder stack.
Save turns red if it is not checked.
The variable that is selected as the scan variable is highlighted in this list. Variables that are overridden from the GUI are also highlight (different color).
Profiles make a number of changes experiment with one select, like enabling or disabling chunks of code. Avoids the list of checkboxes in previous GUI.
Pytheas GUI Experiment Explorer
myExperimentName
Pauseable?
Repeat every 60 sec
Max Pause 40
Num Expts 40
Build
Experiment definition file
Run
C:/qc/exptFiles/SidebandCooling.
Save? Output:
C:/qc/userData/output/
Expt Variables
Value
Scan-able?
arm_t 10.00 Y
final_p 0 Y
pi_time 32.447066 Y
AO pulse shaping H N
Be Separate Wvf Wvf 1.1 N
Profile Select:
DetectionShelving
Sideband Cooling
Doppler Only
Enable Scan
Variable to scan:
arm_t
Start
Stop
Points
Scans
Random?
0
15000
6
1
<-->
CenterOn Cursor
->
Scan history
Parameter
Value
Last Scan
tFlop 15.6 Today 09:35:10
Freq 254 Today 09:40:20
LabBrick 1 -0.265 Yesterday 12:20:20
Data Linker
Plotting
Debug
QuickPlot
X Data
Y Data
Histograms
Scan Variable
PMT1_avg
PMT1_std
NI_AIO2_time
NI_AIO2_avg
PMT1 [ ]
PMT2 [ ]
Fit function:
Choose Y data to fit:
ComboBox
Choose fit function:
y(x) = x^2
y(x) = sin(x)
Refit
Param
Guess
Fitted
Fitted Err
Fixed?
K0 0 0 0
K1 1 1.2 0.05
Analysis file
Filename:
C:/qc/exptFiles/SidebandCooling.dc
Run
A tab for anything related to debugging the sequence of code for the experiment. More ideas for useful GUI things?
Complie Error list
Item One
Item Two
Item Three
Num Errors 3
Num Pulses 40
Pulse Visualizer
Visualize
Visualize Depth 1
Just a place holder graphic
Build is used if edits to the file would cause a change to the GUI, for example when the experiment is first created or if some new kind of tab is added.
Visualize pops open a window to show the experiment pulses. Depth specifies complexity of the graph. Could use flags built into pulses in the experiement code to provide the different levels.
Plotting Tab:
Idea is that Quick plot is only for making 1 simple X vs Y plot (and displaying histograms). Make this DEAD simple. If more complex analysis is to be done, open up the text file below.
In the code, have to define certain variables as data that can be plotted, then it shows up on list.

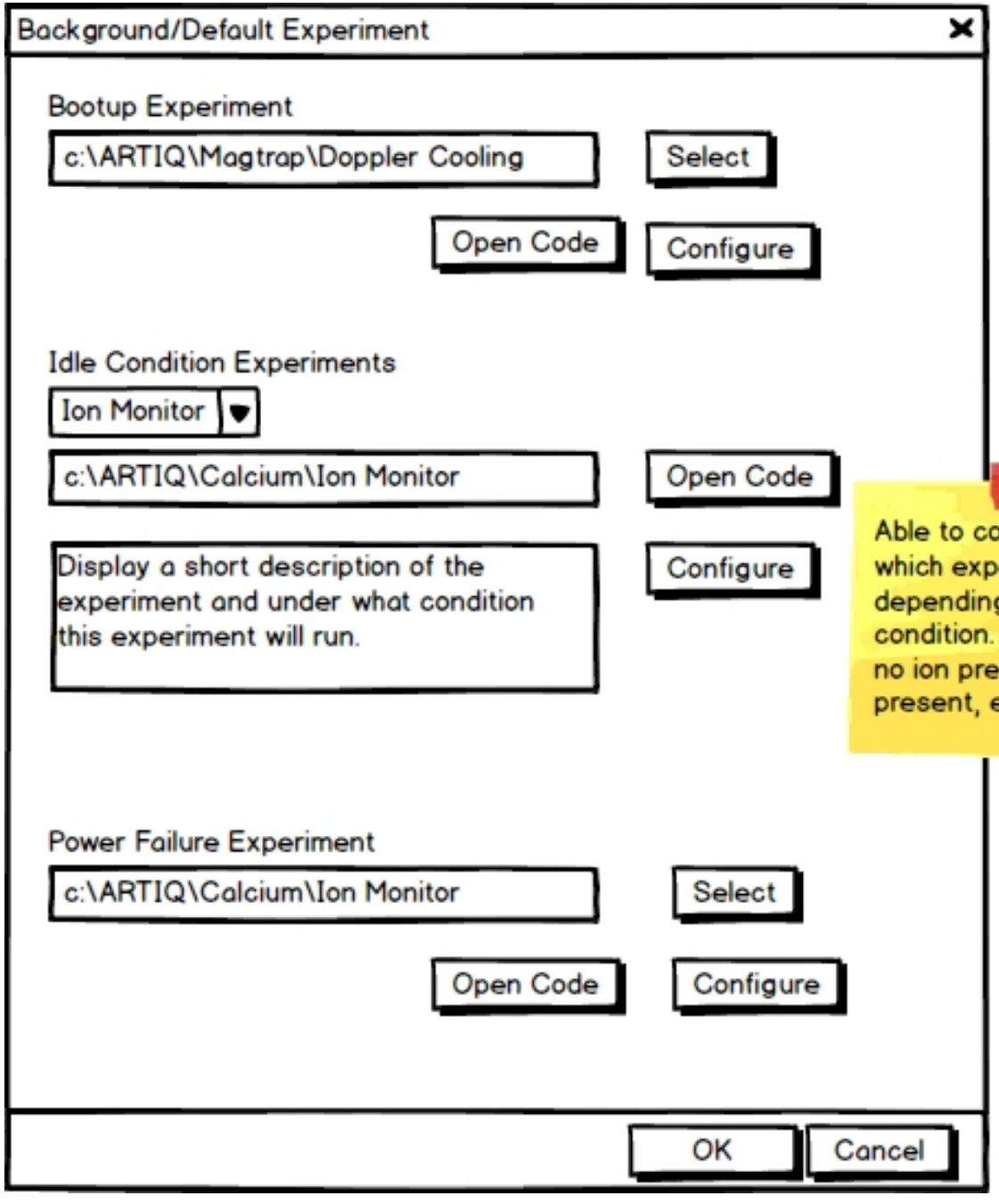

TTL Switch Panel
TTL Grouping
Be Raman All
Be Doppler All
Mg Co
ODF
etc.
Add
Configure
Clear
Be Co
TTL 0
Be 90
TTL 1
uWv 1
TTL 2
BD
TTL 3
Mg 90
TTL 4
ODF
TTL 5
TTL 6
TTL 7
Description of the TTL (User define)
Clicking the button to rotationally force on,off or expt. control. The color on the button show the status of that TTL
The blue/purple indicators shows the current status of the TTL. They are update in real time.
All On
All Off
Expt. Control
Configure
Change description, how the status indicator shows the status of the TTL, etc....



When checked, user can manually override the DDS setting even when the experiment that is running try to change the setting

When left uncheck, the properties is controlled by the running experiment

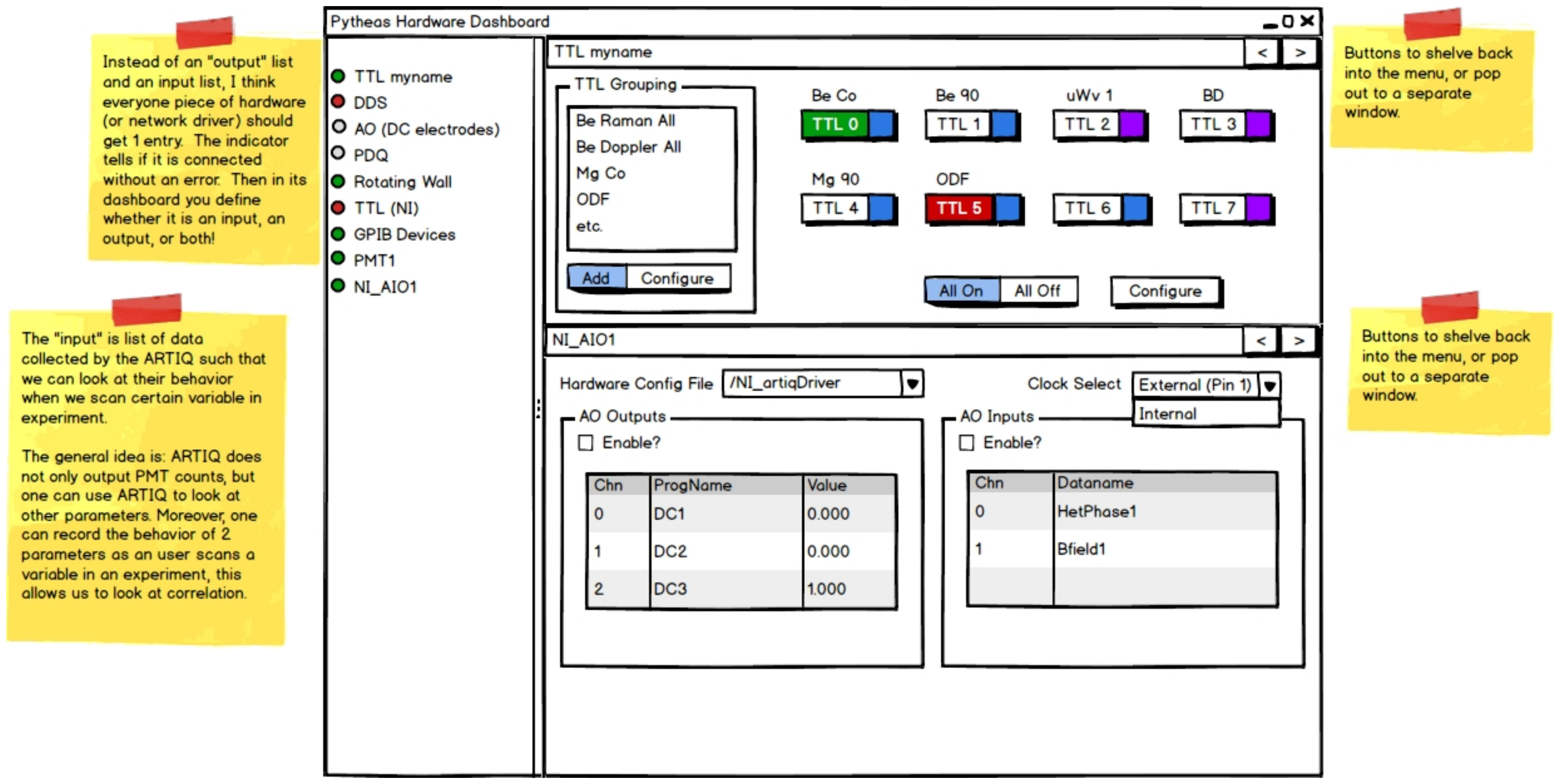
Control the setting of all DDS



Able to configure under which experiment will run depending on the condition. E.g. 10pm-6am, no ion present, ion present, etc.

Clicking on the configure button will pop up the experiment control windows which is similar to the "default" experiment control window, except there will be an obvious indication that this experiment window is setting the experiment control of the Bootup Experiment, the Idle experiments or the Power failure experiment.

In principle, one can even group every "special condition" experiment into a selection of experiment with different given condition to run. The windows on the left will enforce that only one experiment will run in the situation of bootup and power failure.


Instead of an "output" list and an input list, I think everyone piece of hardware (or network driver) should get 1 entry. The indicator tells if it is connected without an error. Then in its dashboard you define whether it is an input, an output, or both!
The "input" is list of data collected by the ARTIQ such that we can look at their behavior when we scan certain variable in experiment.
The general idea is: ARTIQ does not only output PMT counts, but one can use ARTIQ to look at other parameters. Moreover, one can record the behavior of 2 parameters as an user scans a variable in an experiment, this allows us to look at correlation.
Pytheas Hardware Dashboard
TTL myname
DDS
AO (DC electrodes)
PDQ
Rotating Wall
TTL (NI)
GPIB Devices
PMT1
NI_AIO1
TTL myname
< >
TTL Grouping
Be Raman All
Be Doppler All
Mg Co
ODF
etc.
Add
Configure
Be Co
Be 90
uWv 1
BD
TTL 0
TTL 1
TTL 2
TTL 3
Mg 90
ODF
TTL 4
TTL 5
TTL 6
TTL 7
All On
All Off
Configure
Buttons to shelve back into the menu, or pop out to a separate window.
NI_AIO1
< >
Hardware Config File
/NI_artiqDriver
Clock Select
External (Pin 1)
Internal
AO Outputs
Enable?
Chn
ProgName
Value
0
DC1
0.000
1
DC2
0.000
2
DC3
1.000
AO Inputs
Enable?
Chn
Dataname
0
HetPhase1
1
Bfield1
Buttons to shelve back into the menu, or pop out to a separate window.

Pytheas GUI Experiment Control

myExperimentName

Experiment definition file

Run | C:/qc/exptFiles/SidebandCooli

Save? Output: C:/qc/userData/outp

Expt Variables	Value
arm_t	10.00
pi_time	32.447066
Freq	236.23
Shim 38	-0.0365
AO Pulse Shaping	Enable
Be Separate Wvf	Wvf v1.1

Many scan variable in this list will be overlapping with the expt variables.

This box allow user to set choose which variable is scanable and which can only be changed but not scanable.

Scan Variable

Expt Variables
arm_t
final_p
initial_p
middle_p
pi_time

Enable Scan ☑

Number of Scan: 1

Start 0 | Stop 15000 | Interval 0.01 | Random? ☐

<--> | CenterOn Cursor | -><-

Scan history

Parameter	Value	Last Scan
tFlop	15.6	Today 09:35:10
Freq	254	Today 09:40:20
LabBrick 1	-0.265	Yesterday 12:20:20

Simple: 2D Plot / Roll Plot

If Roll Plot: Roll Plot Max time

As a fail safe if something will go wrong at long time.

Measurement X | Plotting X | Quick Control X | Repeat X | Saving X | Analysis X

QuickPlot

Fit Type: Sine Curve (simple) — Lorentzian / Exponential / Parabola

a Sin(x+phi) +c

Show the functional form of the fit function

Parameter	Value	Error
a	0.86	0.002
phi	0.25	0.006
c	0.652	0.02

Ting Rei's notes:

Maybe another window for plotting control and analysis?

The "Analysis" tab is for using customized analysis routine (Python or other software) to analyze the data.

Measurement | Plotting | Quick Control | Repeat | Saving | Analysis | Alerting

- ☑ PMT 1 — Setting
  - X Show Counts
  - X Show Histogram
  - X Update to reference histogram
- ☐ PMT 2 — Setting
  - X Show Counts
  - [ ] Show Histogram
  - [ ] Update to reference histogram
- ☑ ADC Channel 1 (AOM temperature) — Setting
  - X Show Value
- ☑ ADC Channel 2 (Room temperature) — Setting

This is a list of measurement input that can be selected as the plotting (and data saving) dependent variables.

If more than one variable is selected, two plotting graph will pop up as a variable is being scanned.

Measurement | Plotting | Quick Control | Repeat | Saving | Analysis | Alerting

- ☑ Be Doppler Cooling
- ☐ Mg Doppler Cooling
- ---
- ☑ Be Inph Sideband Cooling
- ☐ Be Ooph Sideband Cooling
- ☐ Mg Inph Sideband Cooling
- ☑ Mg Ooph Sideband Cooling
- ---
- ☑ Be Inph 2nd Sideband Cooling
- ☐ Mg Ooph 2nd Sideband Cooling
- ---
- ☑ Detection Shelving
- ☐ Spin Flip before Detection
- ---
- ☐ etc...

Difference:
Experiment variables is changing the state of a certain parameter, the check list is to control if certain feature (e.g. a pulse sequence) will be applied.

A quick checklikst that do quick modification to the experiment, and can change in real time.

This list is not always present in every experiment control window. Depending on different experiemnt, this list will be different. This list of items and what they do to control the experiment will be scripted at the DC file level.

Measurement | Plotting | Quick Control | Repeat | Saving | Analysis | Alerting

- ☐ Pausable
- ☐ Repeat 300s — Setting
- ☐ Automatic Loading if loss ion — Setting
- ☐ Rescue Ion — Setting

This panel shouldn't be called "Repeat", but I don't really have a good name for it right now.

The function of this panel is to put extra feature to the experiment, and these features can be add/remove by user depending on which experiment.

For example: these features can be:
1. "Rescue ion": more far detuned Doppler cooling if the flourescence is below certain threshold
2. "Automatic loading": Load ion automatically in the event of loss ion (this will be very useful for long algorithm experiment such as RB)
3.

Measurement | Plotting | Quick Control | Repeat | Saving | Analysis | Alerting

PMT 1

- ☑ Fluorescence — Setting
  C:/qc/Out/SidebandCooling/PMT1_Fluorescence.csv
- ☑ Histogram — Setting
  C:/qc/Out/SidebandCooling/PMT1_Histogram.csv
- ☐ 2 Be Reference Histrogram Maximum Likelihood Fit — Setting
  C:/qc/Out/SidebandCooling/PMT1_RefHistFit.csv
- ☑ Agnostic Analysis 1 — Setting
  C:/qc/Out/SidebandCooling/PMT1_Agnostic1.csv

ADC Channel 1

- ☑ Temperature — Setting
  C:/qc/Out/SidebandCooling/ADC1_temp.csv

Scenario:

User write an analysis routine to analyze PMT data in realtime. Thus, he will have the raw data: the PMT flourescence and histogram and the analyzed data. The user will have the option to choose which data he want to save for later analysis.

Measurement | Plotting | Quick Control | Repeat | Saving | Analysis | Alerting

Alert me if

- ☑ PMT 1 Fluorescence Count below 20
- ☑ Fit Fail
- ☐ ADC Channel 7 below 0.85
- ☐ TTL Input 4 is High

etc.

Alert Method:

- ☐ Sticky Note on Console
- ☑ Console Update
- ☐ Prompted Window
- ☐ TTL 9 output HIGH
- ☐ Record in Data Save File
- ☑ Email Me tingrei.tan@nist.gov
- ☐ Text Me
- ☐ Clash the GUI

etc.

Configuration

Pytheas GUI Experiment Control

myExperimentName

Experiment definition file

Run | C:/qc/exptFiles/SidebandCooli

Save? Output: C:/qc/userData/outp

Expt Variables	Value
arm_t	10.00
pi_time	32.447066
Freq	236.23
Shim 38	-0.0365
AO Pulse Shaping	Enable
Be Separate Wvf	Wvf v1.1

Scan Variable 1

arm_t (final_p, freq, pi_time, Waveplate 1)

Start: 0 | Stop: 15000 | om?

<--> | CenterOn Cursor | -><-

Scan Variable 2

Waveplate 1

Start: 0 | Stop: 15000 | Interval: 0.01 | Random?

<--> | CenterOn Cursor | -><-

Simple (2D Plot, Roll Plot)

If Roll Plot: Roll Plot Max time

Scan history

Parameter	Value	Last Scan
tFlop	15.6	Today 09:35:10
Freq	254	Today 09:40:20
LabBrick 1	-0.265	Yesterday 12:20:20

Number of Scan: 1

Measurement X | Plotting X | Quick Control X | Repeat X | Saving X | Analysis X

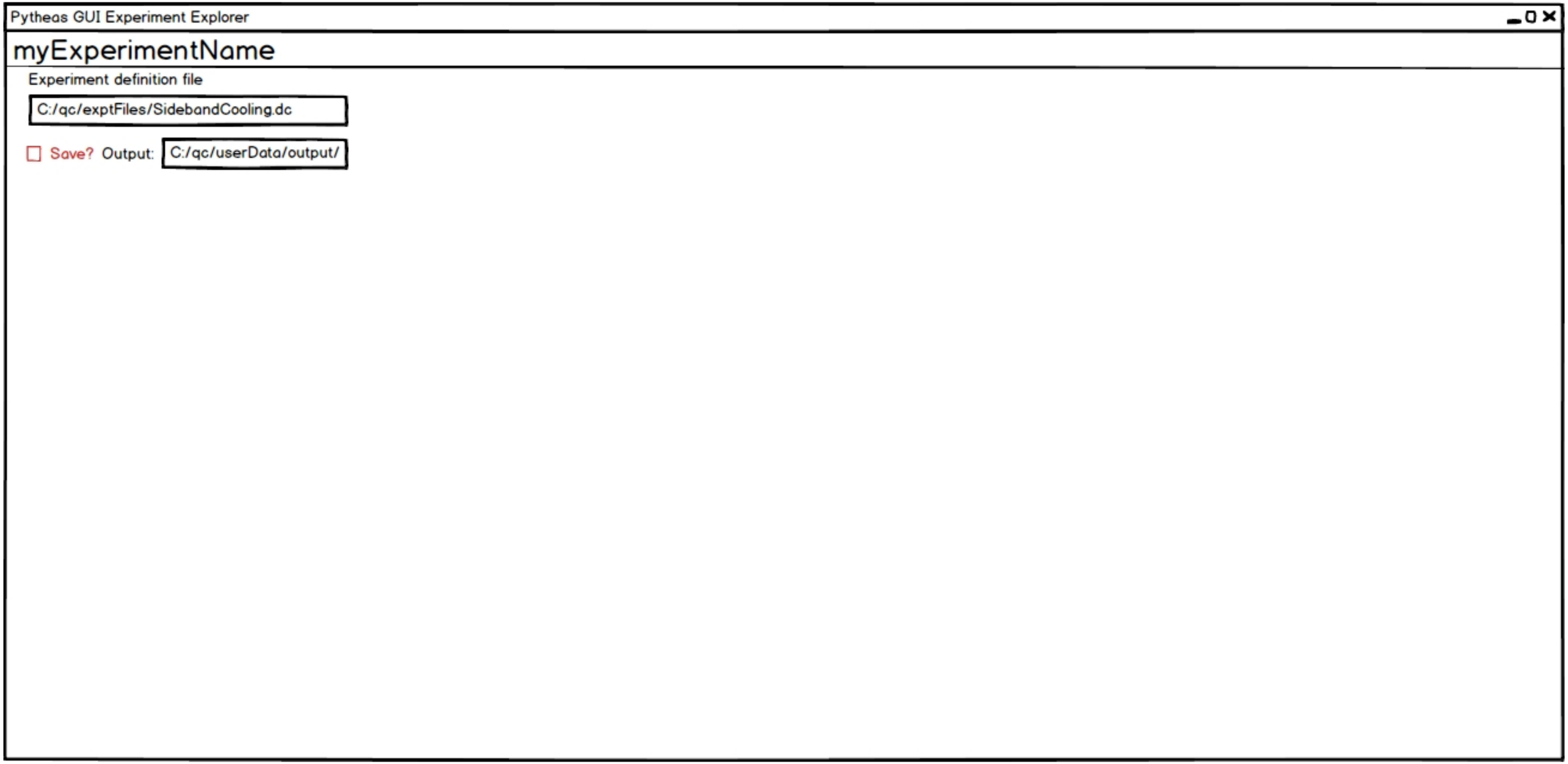
QuickPlot

Fit Type: Sine Curve (simple) (Lorentzian, Exponential, Parabola)

a Sin(x+phi) +c

Show the functional form of the fit function

Parameter	Value	Error
a	0.86	0.002
phi	0.25	0.006
c	0.652	0.02


Pytheas GUI Experiment Explorer
myExperimentName
Experiment definition file
C:/qc/exptFiles/SidebandCooling.dc
Save?
Output:
C:/qc/userData/output/