USING RUNMANAGER

Contents

Version 1.0, SPJ 5 June 2013

Introduction

`runmanager` is a tool used to compile experiment scripts into hardware instructions, and manage global variables ("globals"). Groups of globals can be created and manipulated, for use in experiment scripts written using the `labscript` API.

Creating or loading globals files

For ease of use, globals are stored in HDF5 files, and may be sorted into groups within each file. An experiment may use globals from one or more files, and groups within can be enabled or disabled for each file.

To create a new globals file, open `runmanager` and click on the "+ Groups" tab on the left and click the "New File" button. A file dialog box will appear, where you can select the name and location of your new file. Upon creation, a new row will be added to the list of loaded globals (see Fig. 1) and clicking on the arrow next to its name will reveal the option to add a new group. Clicking this will allow you to type the new group name, and automatically load a tab for that group. Note that the tab can be closed at any time without affecting the globals within it by clicking the red "X" next to the name. Clicking the red "-" will delete that group. Only "active" groups will be used to compile experiments. Groups are activated by ticking the checkbox next to the group name in the list.

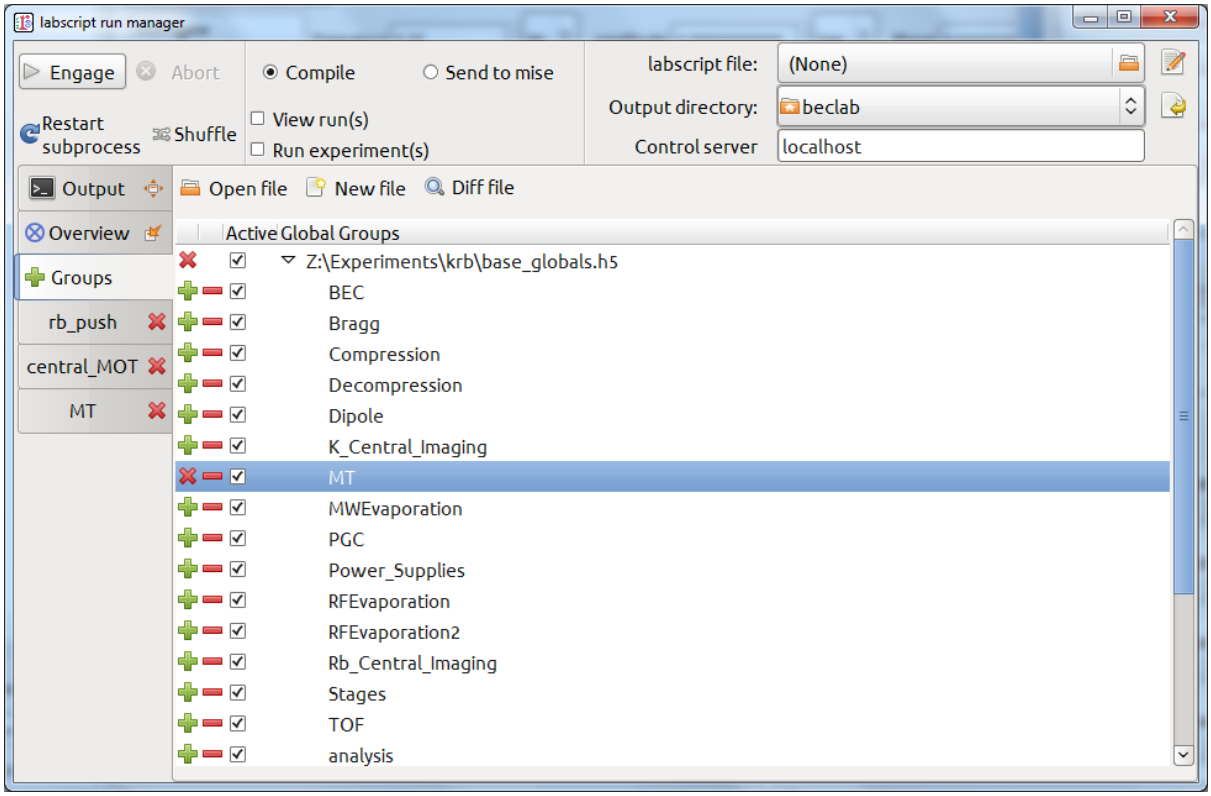

Figure 1 The "+ Groups" tab displays a list of globals files loaded in `runmanager`, and the groups of globals contained in them. Groups marked as "active" will be used in the compilation process. Globals in each group can be manipulated by opening the group's tab by clicking the green "+" next to its name.

Existing globals files can be loaded by clicking "Open file" and selecting your file. A row will be added to the list for the file, and clicking the arrow will reveal the groups in that file. Groups can be activated by checking their boxes and tabs opened for inspecting and manipulating globals for each group by clicking the green "+".

Manipulating globals

Globals can be created and manipulated for each group in its tab (see Fig. 2). To add a global, click the tab for the group you want to add it to, and click "<Click to add new global>". Enter a name for the new global (which corresponds to the name you use when referring to it in your experiment script) and a value. You may type the units corresponding to the global, but note that this is for information only, and will not be used for conversions by your script.

If the value of the global is "True" or "False", the unit will be automatically set to "Bool." and a checkbox will appear for faster switching of the state. The value of a global may be any valid python expression, however note that this will be evaluated by `runmanager`, not your experiment script. The result of the evaluation is shown as a tooltip when hovering over the value. If the output is an iterable type they are used to create a parameter space, as described in the next section (with the only exception being tuples, which are passed as-is).

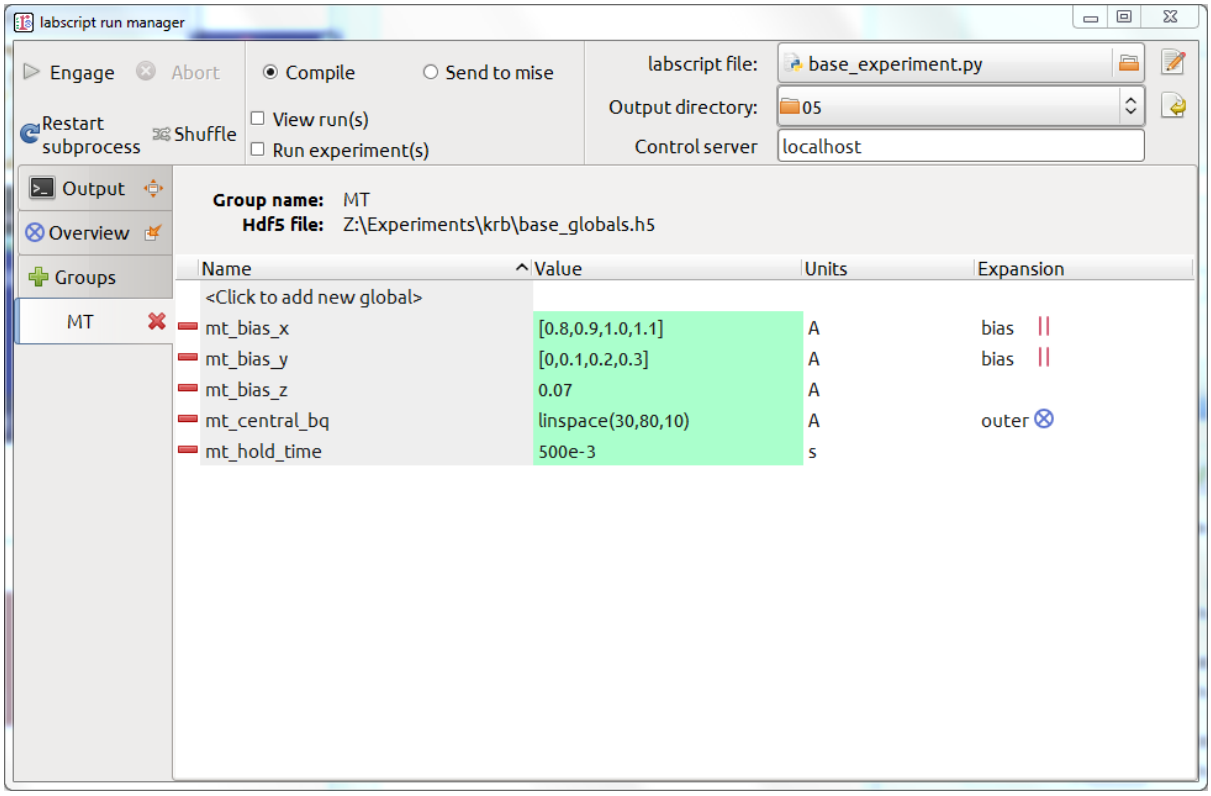

Figure 2 Manipulation of globals. The name entered here is the name used in your python experiment script, and must be a valid python variable name. The value column will turn green upon validation, and a tooltip will show the evaluated expression or error message if something is wrong. The expansion column will indicate if the global belongs to a zip group or will be used in a Cartesian product if the variable contains a list of values. In this case, two of the bias fields, `mt_bias_x` and `mt_bias_y` are being scanned in lock-step. A “zip group” called “bias” has been entered in the expansion column to achieve this. The value of `mt_central_bq` is also being scanned over. A Cartesian product will be performed between these values and the pairs of bias values, in the same way as in the example below.

Parameter spaces

When you enter a list of values for a global (any iterable other than a tuple, e.g. a list, [1,2,3] or an array linspace(0,10,100)), runmanager creates an experiment shot for each value. If multiple values contain lists, the default behaviour is to perform a Cartesian product and produce a shot for each point in the resulting parameter space.

This behaviour can be modified by specifying a "zip group" for a global in the "Expansion" column (by default you will see "outer" here). This will result in globals within a zip group (i.e. globals with the same zip group name) being paired up in lock-step.

If you have multiple zip groups or other non-zipped lists, a Cartesian product will be performed with them, with each zip group treated as though it is a list of tuples.

Note that in future versions of runmanager the “Overview” tab will provide an overview of the parameter space to be scanned, however this feature has not yet been implemented.

Example of using a zip group in conjunction with a non-zipped multi-valued global:

Say we have three globals that contain lists of values, A=[a0,a1,a2], B=[b0,b1,b2] and C=[c0,c1,c2]. If A and B are members of the zip group AB, then when the value of A is a0, B will always be b0 and so on. The resulting parameter space when the Cartesian product is performed with C will contain 9 points (shown as rows in the table):

	Global		
	A	B	C
Value	a0	b0	c0
	a0	b0	c1
	a0	b0	c2
	a1	b1	c0
	a1	b1	c1
	a1	b1	c2
	a2	b2	c0
	a2	b2	c1
	a2	b2	c2

Compiling experiments

Select the experiment file written using the `labscript` API ("`labscript` file") via a file selection dialog box by clicking the icon in the top right hand corner of the screen. For easy access, you can click the edit button next to the file chooser to open the script in your preferred text editor (defined in the experiment configuration file).

The output directory indicates where the compiled HDF5 file for your experiments will be saved. If BLACS, BIAS or `lyse` are running on a different computer then ensure that the files are saved to a shared drive. By default, the output directory will be located in your experiments folder, defined in your experiment configuration file. A folder will be created named after your experiment script, and subfolders for each day, sorted by month. If you manually change the output directory, you can return to the automatic location by clicking the restore button to the right of the selector.

Leaving the radio button in the centre of the top pane on "Compile" (the use of the "Send to `mise`" option will be discussed in the `mise` documentation), clicking the "Engage" button will compile your experiment. The "output" tab will provide information about the compile process, including the output of any print statements in your code and display any errors that may occur. Warnings are also displayed when devices are set to default values due to missing values in your script.

If the "View run(s)" checkbox is enabled your HDF file will open in `runviewer` when it has compiled. If the "Run experiment(s)" checkbox is enabled then the file will be placed in the queue of BLACS, running on the computer identified in the "Control server" field.

When the "Shuffle" button is enabled, shots within a parameter space scan will be shuffled. This can be used to help prevent experimental drift being confused for a variation in a parameter you are scanning.

The “Abort” button will halt compilation. Any file that is currently compiling will finish compiling, but will not be submitted to BLACS. This button is especially useful if you accidentally create a parameter space scan that is much larger than you anticipated!