blacs

Release 0.1.0.dev187+g26cfa05

labscript suite contributors

Feb 09, 2024

DOCUMENTATION

A graphical interface to scientific instruments and experiment supervision.

CHAPTER

ONE

INTRODUCTION

BLACS interfaces with the hardware devices controlling an experiment, and manages the queue of shots to execute on the apparatus. BLACS has two modes of operation: the execution of shots under hardware timing, and the manual control of hardware devices (by the user) via the BLACS GUI. The interface is shown in Fig. 1.1 and is split into two sections that align with the two modes of operation: the queue of shots to execute, and a GUI interface for manually controlling the output state of the hardware devices when not running shots (which can be useful for manual debugging of an apparatus).

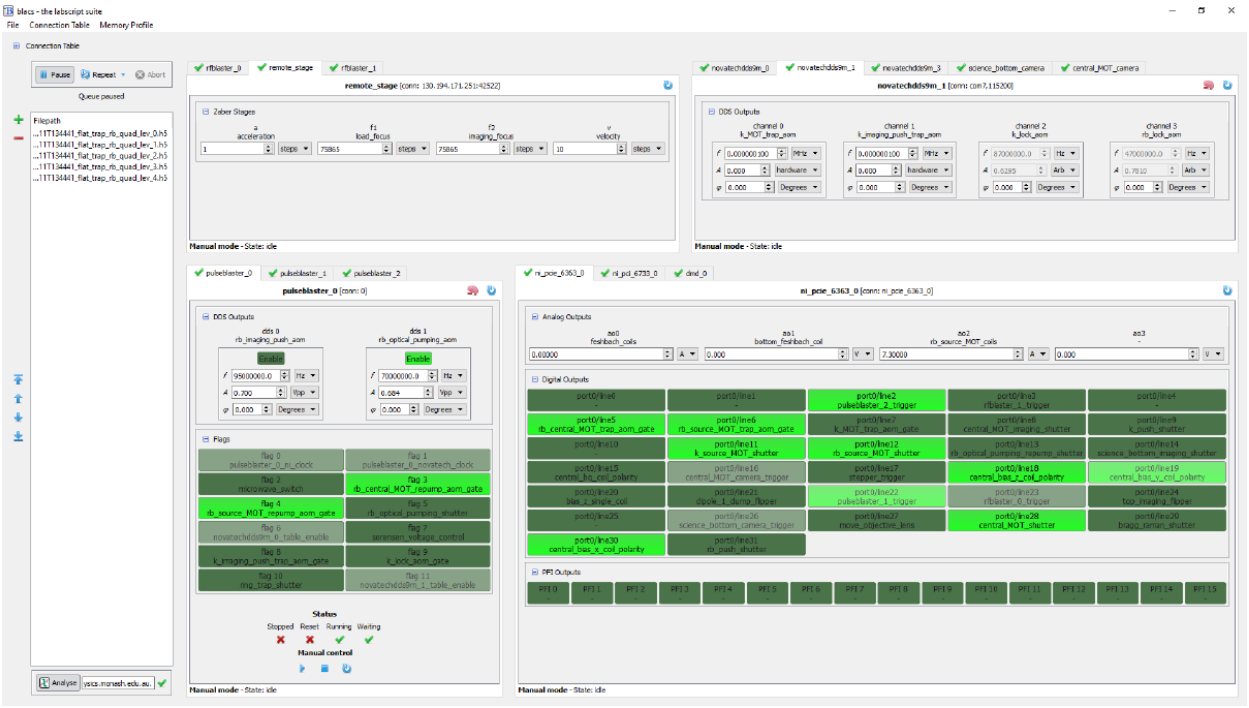

Fig. 1.1: The BLACS interface. Left: The shot queue. Main: A set of tabs (one for each hardware device) that provide a manual control interface for each device. Further details on the BLACS interface can be found in *Usage*.

The shot queue contains standard controls for adding deleting and reordering shots. The queue can also be paused or put into one of several modes that repeat the shots in the queue. When a shot finishes, and the results have been saved to the hdf5 file, the shot may be optionally sent to the lyse server specified in the GUI.

The GUI for each hardware device is dynamically generated at runtime, based on a connection table written using the labscript API. A device tab is created for each device, and communicates with a unique worker process which, in turn, handles the communication with the hardware device. The device tab GUI is populated with controls for each input/output (IO) channel present. To aid in the identification of a relevant I/O channel, controls are labelled in BLACS using both the hardware I/O port name and a user specified name from the connection table of a labscript file. Analog

channels (or more complex output types like a DDS that are represented by several analog numbers) also integrate with the unit conversions specified in the connection table, allowing both control and the display of units other than the (device specific) default hardware unit. Channels connected to sensitive equipment can have the output values limited or the control locked entirely to prevent accidental changes. Output values are stored on exit, and restored on start-up, to avoid unexpected output transients.

CHAPTER

TWO

USAGE

BLACS is the primary interface between experiment shot files created by runmanager, and the hardware devices that control the apparatus. BLACS provides a graphical interface for users to manage the execution of shots, and manually control the output state of hardware devices. In order to support heterogenous hardware, the functionality of BLACS can be extended by developers (who implement support for custom devices) through the provided BLACS API. BLACS thus broadly consists of a set of device code that interfaces with the hardware and provides programmatic and manual control of that hardware, which is discussed in *Device Tabs*, and a shot management routine that receives shot files from runmanager and schedules their execution on the apparatus, which is discussed in *Shot Management*.

BLACS is also readily extensible using a plugin system, discussed in *Plugins*.

2.1 Device Tabs

BLACS creates a tab, in the GUI, for each device it is to control. This information is sourced from a lab connection table, defined using the labscript API, which is kept up to date with the current configuration of hardware in the lab. Much of the BLACS GUI is thus dynamically generated, creating an interface suited to a particular apparatus configuration rather than enforcing a particular style. These tabs encapsulate three components: the code that produces the graphical interface, the worker process(es) (which communicate with the actual hardware), and a state machine which handles communication between the GUI and the worker process(es).

2.1.1 The graphical interface

Each tab GUI is generated from a set of standard components in order to bring uniformity to the control of heterogeneous hardware. This also simplifies the process of adding support for new hardware devices (see How to Add a Device) as the author of the device code does not require knowledge of the GUI widget toolkit. Each tab comprises the following sections (see Fig. 2.1 and Fig. 2.2):

1. device management shortcuts (such as restarting the device),
2. a region (usually hidden) for displaying error messages from the worker process,
3. arrays of 'manual' controls for interacting with each of the device's input and output channels when shots are not running,
4. custom controls specific to a particular device (for example status indicators), and
5. the current state of the state machine (see *State machine*).

The most prominent feature is the arrays of manual controls. These are particularly useful for manual debugging of an experiment apparatus outside of running shots. For easy identification, each channel is automatically named with both the hardware output port, and any assigned name from the lab connection table. All analog values also have an associated dropdown list, where the current unit is displayed. Unit conversions are automatically determined from the lab connection table (where they are defined using the labscript API). This makes debugging simpler as you can

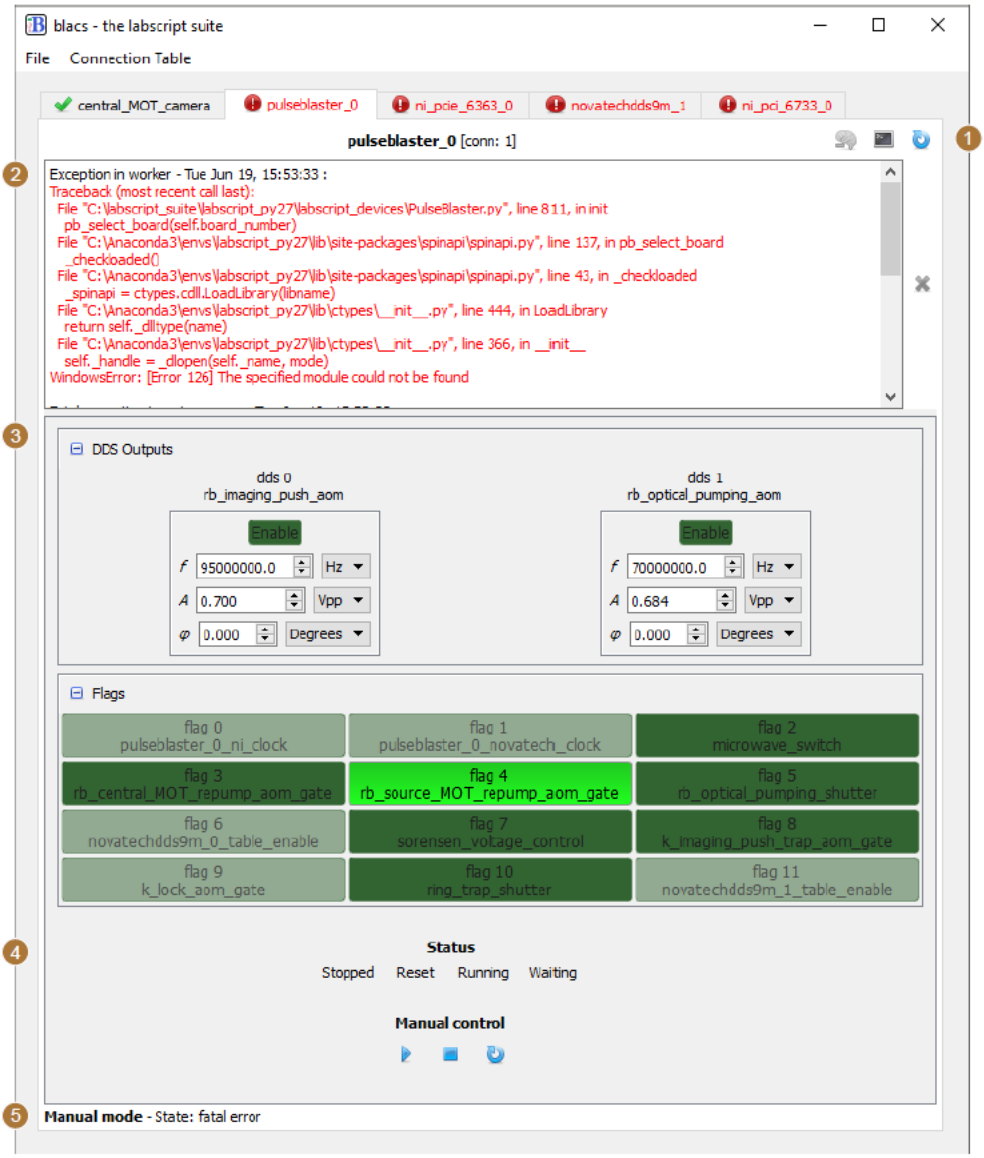

Fig. 2.1: An example of a BLACS tab for a PulseBlaster DDS-II-300-AWG device. The numbered labels match the listing in *The graphical interface*.

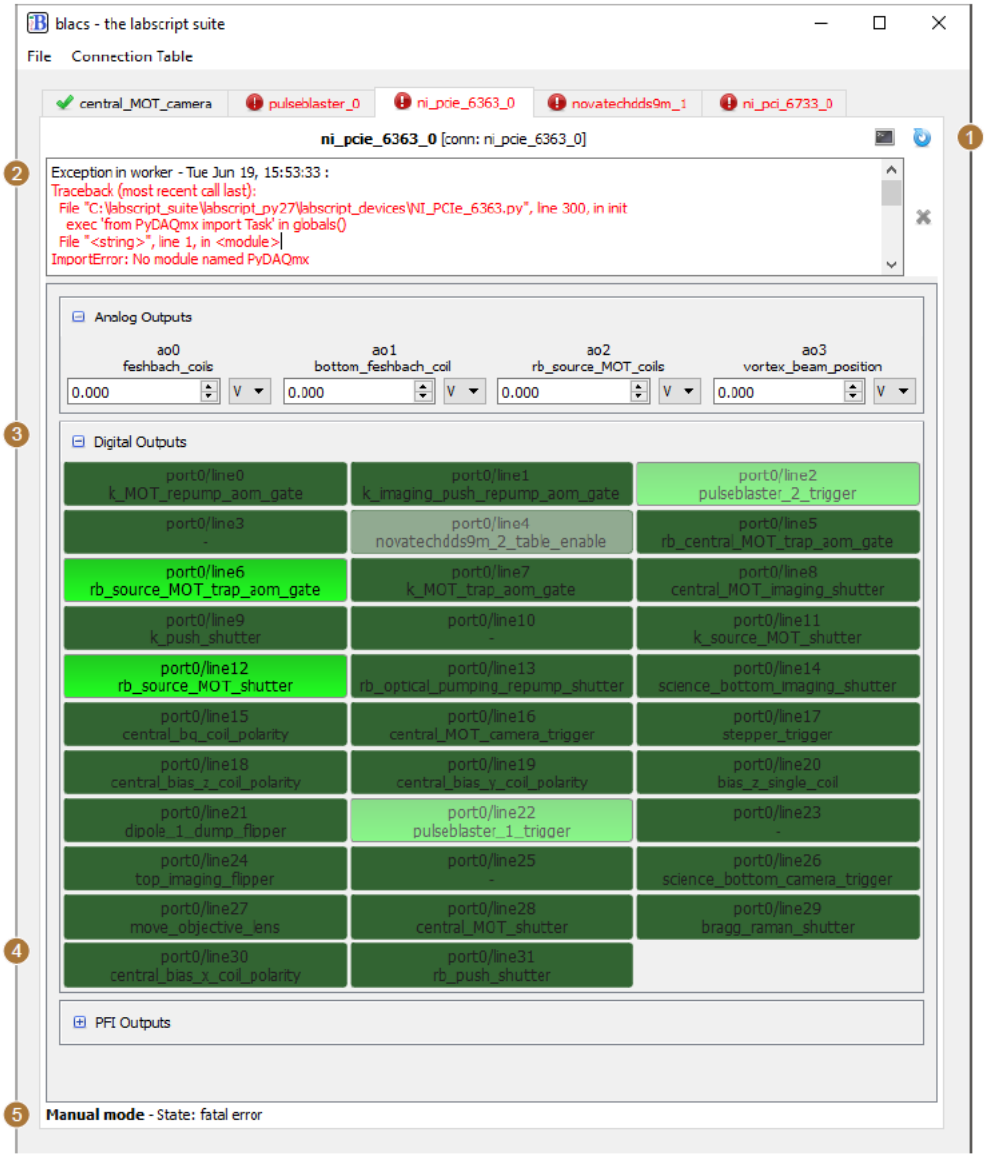

Fig. 2.2: An example of a BLACS tab for an NI PCIe-6363 device. The numbered labels match the listing in *The graphical interface*.

immediately be sure of the output quantity in real world units (for example, the strength of a magnetic field). All output controls can be locked via a right-click context menu to prevent accidental change of their state, which is particularly important when controlling sensitive equipment that can be damaged. For analog quantities, the default step size used when incrementing or decrementing the value[1] can also be customised via the right-click context menu.

The values displayed in the manual controls are also coupled to the hardware device capabilities. The device code that programs the hardware (see worker processes in *Worker processes*) has the ability to return a new value for each channel, each time the device is programmed, allowing the quantised, rounded or coerced value to be returned such that the manual control faithfully displays the output state. BLACS also provides an architecture to periodically poll device values for devices that support such queries. This is particularly important for devices that are not physically restricted to being controlled by a single user (for example, devices controlled via a web interface) or devices that don't remember their state after being power cycled. For such devices, BLACS continually compares the device state with the values displayed in the GUI. If a difference is detected, BLACS presents the user with options to select either the device state or the local state on a per output basis (see Fig. 2.3).

2.1.2 Worker processes

For each device, BLACS launches at least one separate Python process (a worker process) for communicating with the hardware. BLACS communicates with the worker process through our own remote procedure call (RPC) protocol. The python process(es) run a thin wrapper around a specified Python class, which allows the parent process (in this case BLACS) to remotely call methods of the class in the worker process. A method in the worker process is invoked by the tab state machine (see *State machine*), via a message sent over a ZMQ socket. The only task of a worker process is to process any data that is sent to it (via the invocation of one of its methods), interact appropriately with the hardware device(s) it manages, and return any relevant data to the state machine. A third party software library, used to interact with a hardware device (typically provided by a hardware manufacturer), is then only loaded within the isolated worker process. There are several benefits to this 'sandboxing' model. Details on writing the code for a worker process can be found in How to Add a Device.

As previously implied, we have implemented the ability for a BLACS device tab to spawn multiple worker processes. This is particularly useful for devices that handle both inputs and output, and whose API allows these inputs and outputs to be separated and managed by separate processes. An example of such a device is a National Instruments acquisition card such as the NI PCIe-6363. For this device, we spawn three worker processes: the first handles analog and digital outputs, the second handles analog acquisition and the third handles monitoring of a counter in order to measure the lengths of any waits. Multiprocessing also results in a reduction in device programming time prior to the start of an experiment shot. Most device programming is I/O bound (not limited by the processing power of the PC). Simultaneously programming all devices used in a shot thus typically completes in the time it takes to program the longest device (rather than the sum of all programming times for sequential programming).

2.1.3 State machine

One of the major changes in BLACS v2.0 (written and released after our paper[2] was published) was the introduction of a more advanced state machine for each device tab. State machines are an important tool in building complex systems as they enforce a workflow (in this case, for GUI-hardware interaction) which improves the stability of the control system. By using a state machine, we enforce control over what actions can be taken at any given time, improving the robustness of our control software. For example, manual controls on the BLACS front panel should not be able to control hardware devices that are under precision timing while executing a shot. A state machine allows such events to either be discarded or queued until an appropriate time, under a consistent set of easily defined rules.

[1] Incrementing or decrementing the value can be done using the up/down arrows next to the value, the mouse scroll wheel, or the arrow keys on the keyboard. The page up/down keys can also be used, which will adjust the value by 10 times the step size. This is distinct from typing a value directly into the widget, which is not affected by the step size. However both incrementing/decrementing and typing a value in will be equally affected by any quantisation demanded by the hardware device, which we discuss in the following paragraph.

[2] P. T. Starkey, C. J. Billington, S. P. Johnstone, M. Jasperse, K. Helmerson, L. D. Turner, and R. P. Anderson. *A scripted control system for autonomous hardware-timed experiments* Review of Scientific Instruments **84**, 085111 (2013). https://doi.org/10.1063/1.4817213

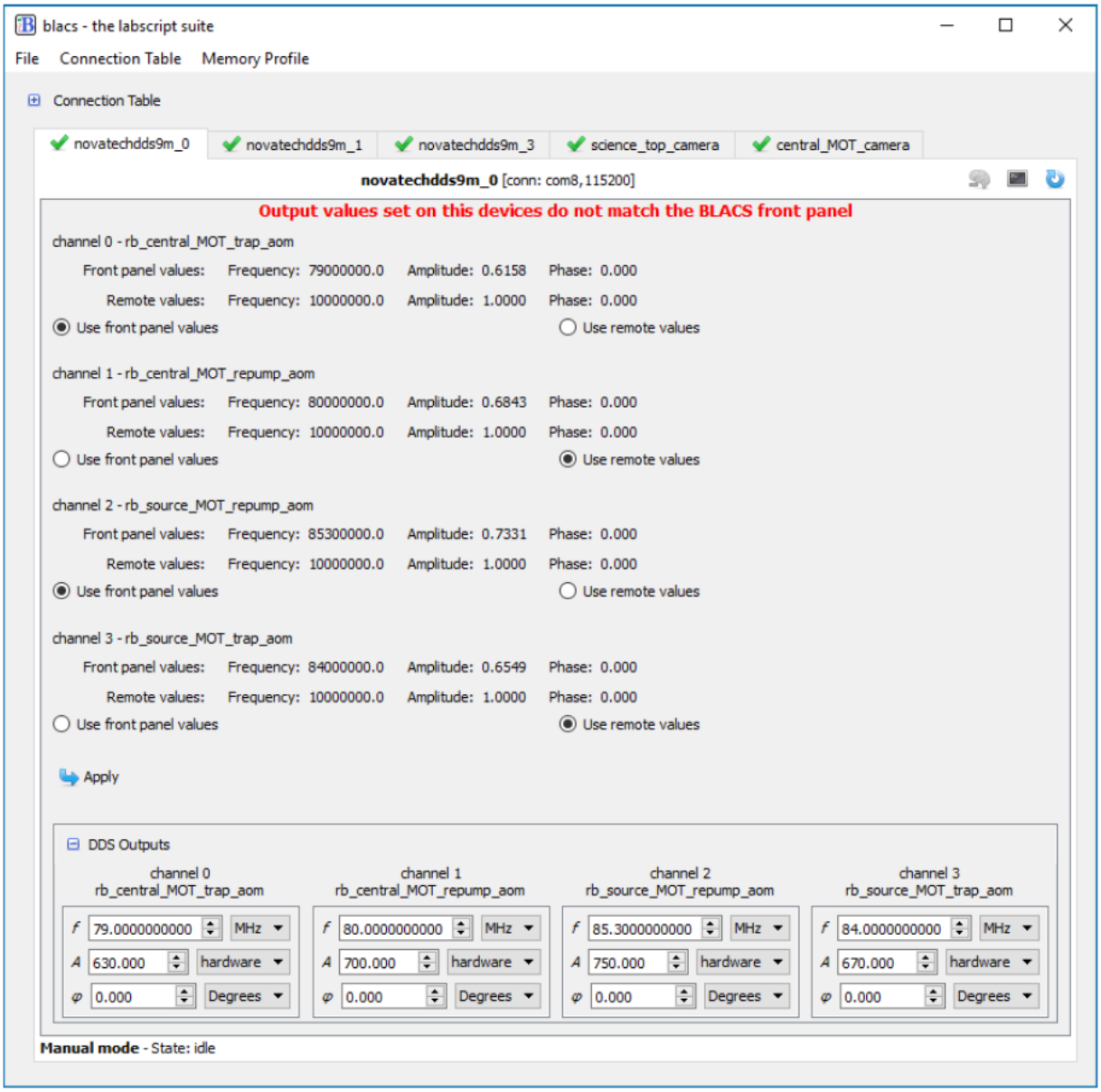

Fig. 2.3: An example showing how devices in BLACS can monitor the consistency between the front panel state and the output of the device (when not running a shot). Here we show a Novatech DDS9m device that has just been power cycled, which causes the output states to reset to a default setting. BLACS detects an inconsistency between the front panel values of BLACS and the output state reported by the device, and presents the GUI pictured above. The user can then to choose either use the local or remote value for each output channel. Once selected, the front panel values of BLACS are updated to the selected value and the device is reprogrammed to match, restoring consistency.

The aim of this state machine is to manage the execution of the device-specific code described previously, which falls into the categories of GUI code and worker-process code. This code exists within Python methods of the device classes, and so will be referred to in this section as the execution of a 'GUI method' and a 'worker process method' respectively. We have implemented a non-standard nested state machine, for which we will coin the term 2D state machine. It consists of two orthogonal sets of states (which we term the inner and outer states) which are linked by the device code. This architecture differs from a standard nested state machine in that it is not hierarchical (events are not passed to the parent state machine as in the hierarchical finite state machine). Our implementation is also unique in that the workflow of the inner (dimension of the) state machine is identical regardless of the outer state.

The outer dimension follows a classical state machine. There are four possible states (which we call modes to distinguish them from the states on the inner dimension):

- mode_manual,
- mode_transition_to_buffered,
- mode_buffered, and
- mode_transition_to_manual.

These four modes represent the two possible modes of operation for the hardware; manual control from BLACS or stepping through instructions during execution of an experiment shot, and the transitions between these modes (where the programming required to change the mode of the device, for example the programming of hardware instructions, usually takes place).

The inner dimension of this two-dimensional state machine is similar to the state machine that existed in BLACS v1.0. There are 5 possible states:

- idle,
- execute (part of) GUI method,
- request execution of worker process method via ZMQ,
- wait for results from worker process method via ZMQ, and
- fatal error.

The inner state machine spends the majority of time in the idle state where it waits for an event to become available from a queue. Events are typically placed in the queue in response to user actions (for example clicking one of the manual control buttons), the 'queue manager' processing a shot (see *Shot Management*), or the timeout of a timer (for example for regular status checks of the hardware).

We define GUI methods that may be queued in the inner state by using a Python decorator, which abstracts away the state machine so that users can call the Python method as normal and be assured that it will always run as part of the state machine (although this enforces the requirement that such methods will return immediately and not return a result to the caller). The decorator itself takes arguments that indicate the modes (of the outer state machine) that the GUI method is allowed to run in, and whether the method should remain in the event queue until the outer state machine enters a mode where it can run, or if it should just be deleted once it reaches the head of the queue. We also provide an option to only run the most recent entry for a method if duplicate entries for the GUI method exist in the queue (albeit with different arguments). This is particularly useful for methods that take a long time to complete but which may be queued up rapidly, for instance a user rapidly changing output values of a device that is slow to program. An example of how you might use the state machine is shown in the definition of a GUI method below.

```
class MyDevice(blacs.device_base_classes.DeviceTab):
    # only run in MODE_MANUAL and keep the state in the queue until
    # the mode condition is met
    @define_state(MODE_MANUAL, True)
    def transition_to_buffered(self, h5_file, notify_queue):
        # set the mode to MODE_TRANSITION_TO_BUFFERED
```

(continues on next page)

(continued from previous page)

```
        self.mode = MODE_TRANSITION_TO_BUFFERED
        # define the set of arguments and keyword arguments
        # to be passed to the worker processes
        args, kwargs = (h5_file,) , {}
        # Yield to the state machine so that the worker process
        # can be run
        result = yield(self.queue_work(self.primary_worker,
            'transition_to_buffered', *args, **kwargs))
        # check that everything was successful ...
        if result :
            # success !
            # update the mode and notify the caller
            self.mode = MODE_BUFFERED
            notify_queue.put([self.device_name, 'success'])
        else:
            # Failure !
            # notify the caller
            notify_queue.put([self.device_name, 'fail'])
            # queue up a method in the state machine
            # to return to MODE_MANUAL and instruct the
            # worker to program the device ready for
            # manual operation
            self.abort_transition_to_buffered()

    @define_state(MODE_TRANSITION_TO_BUFFERED, False )
    def abort_transition_to_buffered(self):
```

This is an example of how one might define the GUI method for triggering the programming of devices so that they are ready for buffered execution of a shot (ready to step through hardware instructions). The GUI method transition_to_buffered has been decorated in order to ensure it is only run as part of the state machine, which means the method will sit in the inner state machine's event queue until the outer state machine mode is set to 'MODE_MANUAL'. When finally executed by the state machine, the method updates the mode of the outer state machine, and yields to the inner state machine in order to tell a worker process to transition into buffered mode (which typically involves programming the table of hardware instructions from the hdf5 shot file). If successful, the outer state machine mode is updated again and the caller of the method (in this case the 'Queue Manager') is notified of the result. If unsuccessful, we call the abort_transition_to_buffered method (which is also decorated as a GUI method) which queues up a new event for the inner state machine. In practice, common functionality like these methods are abstracted away from the user and contained within the blacs.device_base_classes.DeviceTab class. They are implemented in a similar (but more generalised) way to the code shown here. For example transition_to_buffered is actually written to support an arbitrary number of worker processes. Further information on adding support for new devices (and how to use the state machine architecture) is included in How to Add a Device.

The state machine for each device tab runs in its own thread and follows a well defined workflow (also shown graphically in figure 6.4) which can be influenced by the device code. When an event is available in the queue (that can run in the current mode of the outer state machine), the inner state machine transitions to the 'execute GUI method' state, and calls the Python method that was queued up. As this method likely interacts with the GUI, the method is executed in the main thread (via a request to the GUI event loop provided by the Qt widget toolkit). This method executes (temporarily blocking interaction with the GUI) until it either completes, or hits the yield Python keyword. The yield keyword in Python returns control of the program to the calling code (in this case our state machine). The yield keyword also allows the called method to return information to the calling code (for example the data variable would be returned if called as yield(data)). We utilise this to allow the GUI method to request that the inner state machine transition through the inner states relating to the worker process, in this case by calling:

```
yield(self.queue_work('worker_name', 'worker_method_name'))
```

If such a statement is encountered, the inner state machine enters the 'request execution of worker method' state and requests the named worker process execute the named method. It then immediately transitions to the 'wait for results from worker' state. Once results have been received from the worker process (after it has run the worker method), the inner state machine re-enters the 'execute GUI method' state, passing in the results from the worker process as the return value of the yield keyword, and continues with the execution of the GUI method from the point it left. This continues until the execution of the GUI method is complete, where the state machine then enters the 'idle' state again. The exception to this is if a Python Exception is raised in the GUI method, in which case the state machine enters the 'Fatal error' state. The GUI method may also request a change in the outer state machine mode, which then determines which events can be processed when the inner state machine next returns to the 'Idle' state.

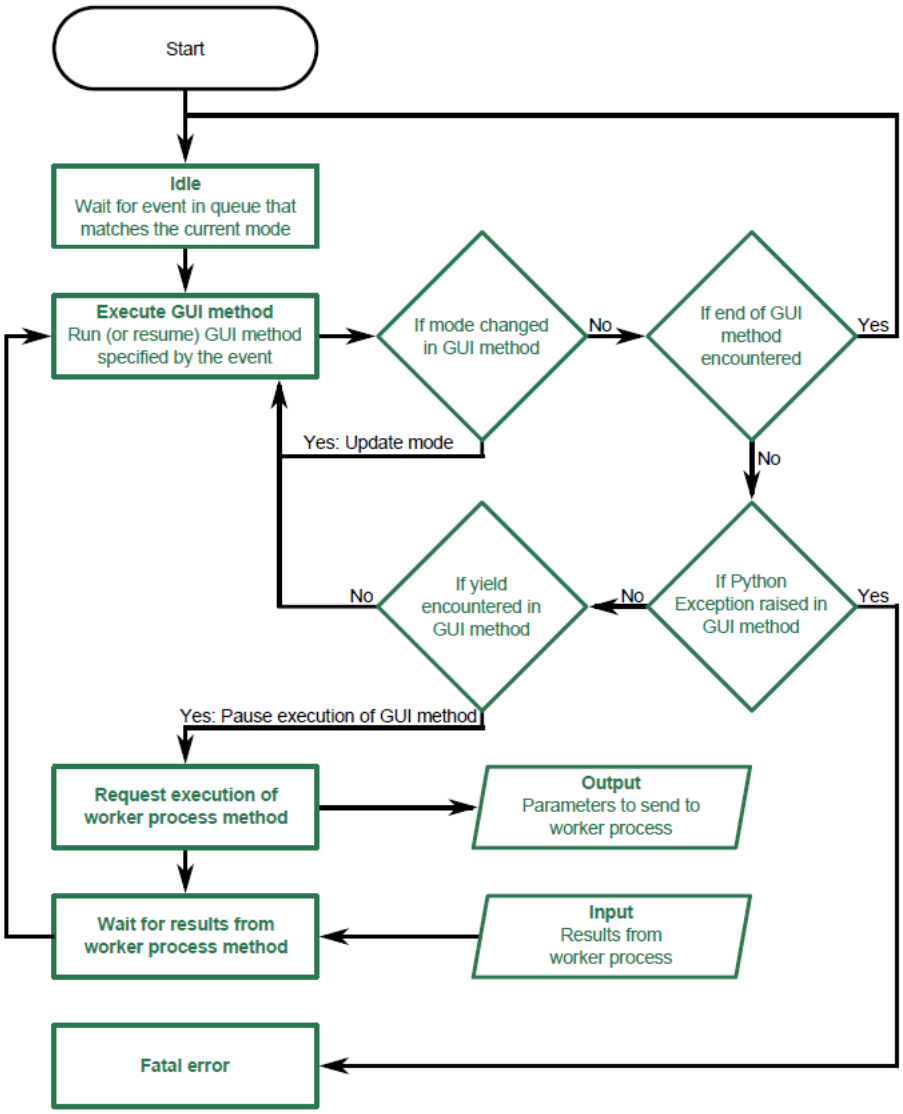

Fig. 2.4: A flowchart of the logic for the BLACS state machine as described in *State machine*.

This results in a flexible state machine architecture that allows the device code to control some portions of the state machine, while maintaining a fixed state machine structure across device tabs. By exposing the internals of the state machine only via the BLACS tab GUI methods, we can abstract away much of the state machine implementation

and simplify the necessary coding skills needed to implement support for new devices. We believe this is a critical requirement of meeting the flexibility goal of our control system, and further details on the simplicity of adding support for new devices is discussed in How to Add a Device. Our state machine architecture also allows us to provide a consistent and responsive GUI to a user by obscuring hardware specific details and offloading these to a separate process.

2.1.4 Handling waits

In order to use waits, one of the devices in BLACS must monitor the length of each wait so that a mapping between labscript time and experiment time can be made. This mapping is then used by analog acquisition devices in BLACS to correctly break up acquired traces into the requested acquisitions. The length of the wait is also used in real (software) time by the wait monitor in order to ensure the experiment is not stuck in a wait forever.

The current wait monitor implementation uses one of the inbuilt counters that are in many National Instruments acquisition devices, however other implementations are of course possible if support is added when creating the device classes. At the completion of an experiment shot, the wait monitor calculates the durations of each wait (based on data it acquired during the shot) and writes these to the shot file. The wait monitor then broadcasts a ZMQ event indicating this has been completed. Device code that relies on the wait information (for example, for breaking up analog acquisitions into the requested set of traces), waits for this event to be received during the transition to manual mode, before performing any action. This ensures that the measured lengths of the waits are always available in the hdf5 file when required.

2.2 Shot Management

The primary purpose of BLACS is to execute experiment shots on the lab apparatus. File paths to shots are typically received by BLACS over ZMQ, but can also be loaded directly through the BLACS GUI (either via the file menu, or by dragging and dropping onto the queue). Prior to accepting the shot, BLACS compares the connection table of the shot to the lab connection table and ensures that the shot is compatible with the current configuration of the laboratory hardware. Connection table compatibility requires that the shot connection table is a subset of the lab connection table. This ensures that old experiments can not be run on hardware that is no-longer configured correctly, preventing damage or unexpected results. Shots that pass this check are added to a queue, which is visible in the BLACS GUI (see Fig. 2.5).

The queue is processed by a thread in BLACS, which we term the 'queue manager', that takes the top-most shot in the queue and, in turn, executes it. Shot execution follows the following pattern (a flowchart of this process is also shown in Fig. 2.6):

1. For each device in use in the shot, a message is sent to the corresponding device tab state machine indicating that the device should program the device for hardware timed execution of a shot. These messages are sent asynchronously, which ensures devices program in parallel if possible (subject to the state machine being available to process the message). Included in this message is the path to the hdf5 shot file, which each device tab ultimately passes to a worker process that in turn, reads out the hardware instructions and programs the hardware. During this programming, the device tab enters the mode 'transition_to_buffered' (see §6.1.1.3).
2. The queue manager then waits until all devices have reported they have programmed, at which point all device tabs in use should be in the 'buffered' state machine mode. If a device does not report it has completed within a 5 minute timeout, or a device reports an error has occurred during programming, the queue manager aborts the shot by pausing the queue, instructing all device tabs to abort, and replacing the shot at the top of the queue.
3. Provided all devices report they are ready, the queue manager proceeds with starting the shot. This involves recording the current state of all manual controls (as these usually affect the initial values of the shot and may affect results in certain experiments) and then instructing the master pseudoclock to begin execution of the programmed instructions.

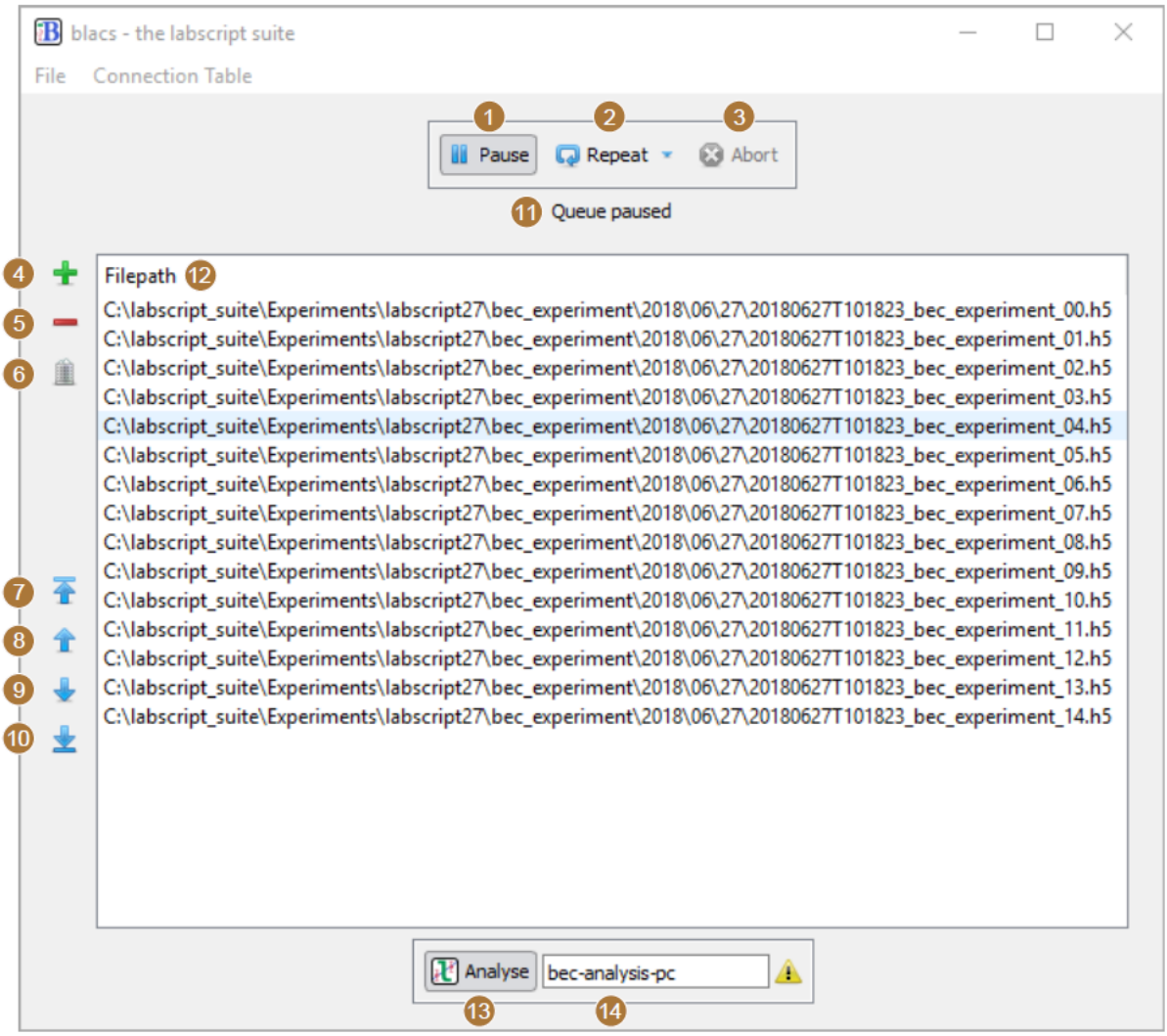

Fig. 2.5: The queue manager GUI within BLACS (with the device tabs hidden). (1) The pause button stops the queue from processing new shots (a currently running shot will finish). (2) The repeat button, when enabled, will duplicate a completed shot and either place the duplicate at the bottom or the top of the queue (depending on the mode selected). (3) The abort button immediately stops the execution of the current shot and returns hardware to manual mode. (4-5) Buttons to add or delete selected shots from the queue. (6) A button to clear the entire queue. (7-10) Buttons to reorder selected shots within the queue. (11) The current status of the queue is displayed here. For example, the status may indicate that devices are currently being programmed, the master pseudoclock has been triggered and the experiment is running, or that acquired data is currently being saved into the hdf5 shot file. (12) The list of shot files in the queue, in the order they will be executed (the topmost is executed first). (13) A button to enable or disable the forwarding of shots to lyse for analysis. (14) The network hostname of the PC running lyse.

4. The queue manager then waits for the master pseudoclock to report that the experiment shot has completed. If an error occurs in a device tab during a shot, the queue manager aborts the shot (as previously described) and pauses the queue.
5. Once a shot has completed, the queue manager instructs all device tabs to 'transition to manual' mode. At this stage, device tabs enter the 'transition_to_manual' state machine mode where they save any acquired data and reprogram the hardware device for manual operation via the BLACS GUI. Again, if errors occur during this process, the queue manager aborts the shot as before, but with the additional step of cleaning any saved data from the hdf5 file (so that the shot file is returned to the state prior to execution).
6. The path to the shot file is now sent to a separate thread that runs a routine for managing submission of shots to lyse for analysis. This routine forwards the shot file paths to the lyse server specified in the BLACS GUI if analysis submission is enabled (see figure 6.5 (13–14)). If lyse does not respond to these messages, the shot file paths are buffered until such time as lyse does respond, to ensure no shots are missing from analysis.
7. Finally, the queue manager checks the state of the repeat button in the BLACS GUI and, if required, duplicates the shot (minus the acquired data) and places the duplicate in the appropriate place in the queue.

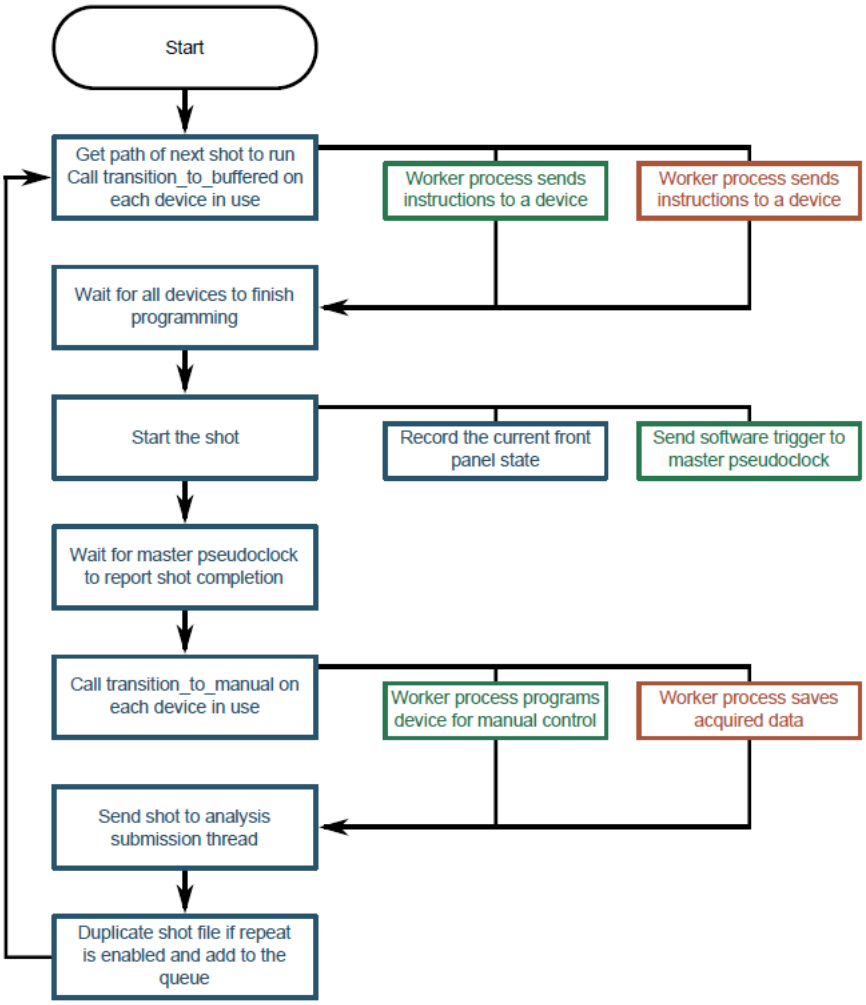

Fig. 2.6: A flowchart of the logic for the BLACS queue manager. For brevity, we have not included the logic for pausing the queue via the GUI or handling error conditions. See the listing above for further details.

2.3 Plugins

To cater to the variety of lab environments, BLACS supports custom user code (that is not covered by a device implementation) in the form of plugins. Plugins allow custom graphical interfaces to be created through the addition of menu items, notifications, preferences editable through a common preferences panel, and custom tabs that sit alongside device tabs. Plugins are also provided access to a variety of internal BLACS objects, such as the shot queue, and can register callback functions to be run when certain events happen (such as a shot completing). This provides a powerful basis for customising the behaviour of BLACS in a way that is both modular and maintainable, providing a way to include optional conflicting features without needing to resolve the incompatibility. Plugins can be easily shared between groups, allowing for a diverse variety of control system interfaces that are all built on a common platform. We have developed several plugins at Monash, which are detailed in the following sections, and demonstrate the broad applicability of the plugin system.

The API refrence for the standard plugins is *here*

2.3.1 The connection table plugin

This plugin is included in the default install of BLACS, and provides a clean interface to manage the lab connection table that BLACS uses to automatically generate the device tabs and their graphical interfaces. The plugin inserts a menu item that provides shortcuts for:

1. editing the connection table Python file,
2. initiating the recompilation of the connection table, and
3. editing the preferences that control the behaviour of the plugin.

The preferences panel allows you to configure a list of hdf5 files containing runmanager globals to use during the compilation of the connection table (commonly used as unit conversion parameters), as well as a list of unit conversion Python files, to watch for any changes. At startup, the plugin launches a background thread that monitors changes to these files, as well as the connection table Python file and compiled connection table hdf5 file. If any modifications are detected, a notification is shown at the top of BLACS informing that the lab connection table should be recompiled. If recompilation is chosen by the user, the plugin manages the recompilation of the connection table using the runmanager API and output from this process is displayed in a window. Once recompilation is successful, the plugin relaunches BLACS so that the new connection table is loaded. This ensures that BLACS is using the same knowledge of the experiment apparatus as any future shots will when they are created by runmanager (assuming they share the globals files used).

CHAPTER

THREE

API REFERENCE

blacs.analysis_submission	
blacs.compile_and_restart	
blacs.device_base_class	
blacs.experiment_queue	
blacs.front_panel_settings	
blacs.notifications	
blacs.output_classes	
blacs.plugins	
blacs.tab_base_classes	
blacs.__main__	BLACS GUI and supporting code

3.1 blacs.analysis_submission

Classes

AnalysisSubmission(BLACS, blacs_ui)	

3.1.1 blacs.analysis_submission.AnalysisSubmission

class blacs.analysis_submission.**AnalysisSubmission**(*BLACS*, *blacs_ui*)

Bases: object

__init__(*BLACS*, *blacs_ui*)

Methods

__init__(BLACS, blacs_ui)
check_connectivity()
check_retry()
clear_waiting_files()
get_queue()
get_save_data()
mainloop()
restore_save_data(data)
submit_waiting_files()
update_waiting_files_message()

Attributes

send_to_server
server
server_online

check_connectivity()

check_retry()

clear_waiting_files()

get_queue()

get_save_data()

mainloop()

restore_save_data(*data*)

property send_to_server

property server

property server_online

submit_waiting_files()

update_waiting_files_message()

3.2 blacs.compile_and_restart

Classes

CompileAndRestart(blacs, globals_files, ...)

3.2.1 blacs.compile_and_restart.CompileAndRestart

class blacs.compile_and_restart.**CompileAndRestart**(*blacs*, *globals_files*, *connection_table_labscript*, *output_path*, *close_notification_func=None*)

Bases: QDialog

__init__(*blacs*, *globals_files*, *connection_table_labscript*, *output_path*, *close_notification_func=None*)

Methods

__init__(blacs, globals_files, ...[, ...])
accept(self)
acceptDrops(self)
accessibleDescription(self)
accessibleName(self)
actionEvent(self, a0)
actions(self)
activateWindow(self)
addAction(self, action)

continues on next page

Table 3.1 – continued from previous page

addActions(self, actions)
adjustSize(self)
autoFillBackground(self)
backgroundRole(self)
baseSize(self)
blockSignals(self, b)
changeEvent(self, a0)
childAt(-> Optional[QWidget])
childEvent(self, a0)
children(self)
childrenRect(self)
childrenRegion(self)
clearFocus(self)
clearMask(self)
close(self)
closeEvent(self, a0)
colorCount(self)
compile()
connectNotify(self, signal)
contentsMargins(self)
contentsRect(self)
contextMenuEvent(self, a0)
contextMenuPolicy(self)
create(self[, window, initializeWindow, ...])
createWindowContainer(window[, parent, flags])
cursor(self)

continues on next page

Table 3.1 – continued from previous page

customEvent(self, a0)
deleteLater(self)
depth(self)
destroy(self[, destroyWindow, destroySubWindows])
devType(self)
devicePixelRatio(self)
devicePixelRatioF(self)
devicePixelRatioFScale()
disconnect(-> bool)
disconnectNotify(self, signal)
done(self, a0)
dragEnterEvent(self, a0)
dragLeaveEvent(self, a0)
dragMoveEvent(self, a0)
dropEvent(self, a0)
dumpObjectInfo(self)
dumpObjectTree(self)
dynamicPropertyNames(self)
effectiveWinId(self)
ensurePolished(self)
enterEvent(self, a0)
event(self, a0)
eventFilter(self, a0, a1)
exec(self)
exec_(self)
find(a0)

continues on next page

Table 3.1 – continued from previous page

findChild(-> QObjectT)
findChildren(...)
finished_compiling(success)
focusInEvent(self, a0)
focusNextChild(self)
focusNextPrevChild(self, next)
focusOutEvent(self, a0)
focusPolicy(self)
focusPreviousChild(self)
focusProxy(self)
focusWidget(self)
font(self)
fontInfo(self)
fontMetrics(self)
foregroundRole(self)
frameGeometry(self)
frameSize(self)
geometry(self)
getContentsMargins(self)
grab(self[, rectangle])
grabGesture(self, type[, flags])
grabKeyboard(self)
grabMouse()
grabShortcut(self, key[, context])
graphicsEffect(self)
graphicsProxyWidget(self)

continues on next page

Table 3.1 – continued from previous page

hasFocus(self)
hasHeightForWidth(self)
hasMouseTracking(self)
hasTabletTracking(self)
height(self)
heightForWidth(self, a0)
heightMM(self)
hide(self)
hideEvent(self, a0)
inherits(self, classname)
initPainter(self, painter)
inputMethodEvent(self, a0)
inputMethodHints(self)
inputMethodQuery(self, a0)
insertAction(self, before, action)
insertActions(self, before, actions)
installEventFilter(self, a0)
isActiveWindow(self)
isAncestorOf(self, child)
isEnabled(self)
isEnabledTo(self, a0)
isFullScreen(self)
isHidden(self)
isLeftToRight(self)
isMaximized(self)
isMinimized(self)

continues on next page

Table 3.1 – continued from previous page

isModal(self)
isRightToLeft(self)
isSignalConnected(self, signal)
isSizeGripEnabled(self)
isVisible(self)
isVisibleTo(self, a0)
isWidgetType(self)
isWindow(self)
isWindowModified(self)
isWindowType(self)
keyPressEvent(self, a0)
keyReleaseEvent(self, a0)
keyboardGrabber()
killTimer(self, id)
layout(self)
layoutDirection(self)
leaveEvent(self, a0)
locale(self)
logicalDpiX(self)
logicalDpiY(self)
lower(self)
mapFrom(self, a0, a1)
mapFromGlobal(self, a0)
mapFromParent(self, a0)
mapTo(self, a0, a1)
mapToGlobal(self, a0)

continues on next page

Table 3.1 – continued from previous page

mapToParent(self, a0)
mask(self)
maximumHeight(self)
maximumSize(self)
maximumWidth(self)
metaObject(self)
metric(self, a0)
minimumHeight(self)
minimumSize(self)
minimumSizeHint(self)
minimumWidth(self)
mouseDoubleClickEvent(self, a0)
mouseGrabber()
mouseMoveEvent(self, a0)
mousePressEvent(self, a0)
mouseReleaseEvent(self, a0)
move()
moveEvent(self, a0)
moveToThread(self, thread)
nativeEvent(self, eventType, message)
nativeParentWidget(self)
nextInFocusChain(self)
normalGeometry(self)
objectName(self)
on_activate_default(window)
open(self)

continues on next page

Table 3.1 – continued from previous page

overrideWindowFlags(self, type)	
overrideWindowState(self, state)	
paintEngine(self)	
paintEvent(self, a0)	
paintingActive(self)	
palette(self)	
parent(self)	
parentWidget(self)	
physicalDpiX(self)	
physicalDpiY(self)	
pos(self)	
previousInFocusChain(self)	
property(self, name)	
pyqtConfigure(...)	Each keyword argument is either the name of a Qt property or a Qt signal.
raise_(self)	
receivers(self, signal)	
rect(self)	
reject(self)	
releaseKeyboard(self)	
releaseMouse(self)	
releaseShortcut(self, id)	
removeAction(self, action)	
removeEventFilter(self, a0)	
render(, sourceRegion, flags, ...)	
repaint(-> None -> None)	
resize()	

continues on next page

Table 3.1 – continued from previous page

resizeEvent(self, a0)
restart()
restoreGeometry(self, geometry)
result(self)
saveGeometry(self)
screen(self)
scroll()
sender(self)
senderSignalIndex(self)
setAcceptDrops(self, on)
setAccessibleDescription(self, description)
setAccessibleName(self, name)
setAttribute(self, attribute[, on])
setAutoFillBackground(self, enabled)
setBackgroundRole(self, a0)
setBaseSize()
setContentsMargins()
setContextMenuPolicy(self, policy)
setCursor(self, a0)
setDisabled(self, a0)
setEnabled(self, a0)
setFixedHeight(self, h)
setFixedSize()
setFixedWidth(self, w)
setFocus()
setFocusPolicy(self, policy)

continues on next page

Table 3.1 – continued from previous page

setFocusProxy(self, a0)
setFont(self, a0)
setForegroundRole(self, a0)
setGeometry()
setGraphicsEffect(self, effect)
setHidden(self, hidden)
setInputMethodHints(self, hints)
setLayout(self, a0)
setLayoutDirection(self, direction)
setLocale(self, locale)
setMask()
setMaximumHeight(self, maxh)
setMaximumSize()
setMaximumWidth(self, maxw)
setMinimumHeight(self, minh)
setMinimumSize()
setMinimumWidth(self, minw)
setModal(self, modal)
setMouseTracking(self, enable)
setObjectName(self, name)
setPalette(self, a0)
setParent()
setProperty(self, name, value)
setResult(self, r)
setShortcutAutoRepeat(self, id[, enabled])
setShortcutEnabled(self, id[, enabled])

continues on next page

Table 3.1 – continued from previous page

setSizeGripEnabled(self, a0)
setSizeIncrement()
setSizePolicy()
setStatusTip(self, a0)
setStyle(self, a0)
setStyleSheet(self, styleSheet)
setTabOrder(a0, a1)
setTabletTracking(self, enable)
setToolTip(self, a0)
setToolTipDuration(self, msec)
setUpdatesEnabled(self, enable)
setVisible(self, visible)
setWhatsThis(self, a0)
setWindowFilePath(self, filePath)
setWindowFlag(self, a0[, on])
setWindowFlags(self, type)
setWindowIcon(self, icon)
setWindowIconText(self, a0)
setWindowModality(self, windowModality)
setWindowModified(self, a0)
setWindowOpacity(self, level)
setWindowRole(self, a0)
setWindowState(self, state)
setWindowTitle(self, a0)
sharedPainter(self)
show(self)

continues on next page

Table 3.1 – continued from previous page

showEvent(self, a0)
showFullScreen(self)
showMaximized(self)
showMinimized(self)
showNormal(self)
signalsBlocked(self)
size(self)
sizeHint(self)
sizeIncrement(self)
sizePolicy(self)
stackUnder(self, a0)
startTimer(self, interval[, timerType])
statusTip(self)
style(self)
styleSheet(self)
tabletEvent(self, a0)
testAttribute(self, attribute)
thread(self)
timerEvent(self, a0)
toolTip(self)
toolTipDuration(self)
tr(self, sourceText[, disambiguation, n])
underMouse(self)
ungrabGesture(self, type)
unsetCursor(self)
unsetLayoutDirection(self)

continues on next page

Table 3.1 – continued from previous page

unsetLocale(self)
update(-> None -> None)
updateGeometry(self)
updateMicroFocus(self)
updatesEnabled(self)
visibleRegion(self)
whatsThis(self)
wheelEvent(self, a0)
width(self)
widthMM(self)
winId(self)
window(self)
windowFilePath(self)
windowFlags(self)
windowHandle(self)
windowIcon(self)
windowIconText(self)
windowModality(self)
windowOpacity(self)
windowRole(self)
windowState(self)
windowTitle(self)
windowType(self)
x(self)
y(self)

Attributes

Accepted	
DrawChildren	
DrawWindowBackground	
IgnoreMask	
PdmDepth	
PdmDevicePixelRatio	
PdmDevicePixelRatioScaled	
PdmDpiX	
PdmDpiY	
PdmHeight	
PdmHeightMM	
PdmNumColors	
PdmPhysicalDpiX	
PdmPhysicalDpiY	
PdmWidth	
PdmWidthMM	
Rejected	
accepted	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
customContextMenuRequested	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
destroyed	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
finished	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
objectNameChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
rejected	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
staticMetaObject	
windowIconChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
windowIconTextChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
windowTitleChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL

closeEvent(*self*, *a0: QCloseEvent | None*)

compile()

finished_compiling(*success*)

on_activate_default(*window*)

restart()

3.3 blacs.device_base_class

Classes

DeviceTab(notebook, settings[, restart])	
DeviceWorker(*args, **kwargs)	

3.3.1 blacs.device_base_class.DeviceTab

class blacs.device_base_class.**DeviceTab**(*notebook*, *settings*, *restart=False*)

Bases: *Tab*

__init__(*notebook*, *settings*, *restart=False*)

Methods

__init__(notebook, settings[, restart])	
abort_buffered(*args, **kwargs)	
abort_transition_to_buffered(*args, **kwargs)	
add_secondary_worker(worker)	
auto_create_widgets()	
auto_place_widgets(*args)	
check_remote_values(*args, **kwargs)	
check_time()	
clean_ui_on_restart()	
close_tab([finalise])	Close the tab, terminate subprocesses and join the mainloop thread.

continues on next page

Table 3.2 – continued from previous page

connect_restart_receiver(function)	
continue_restart(currentpage)	Called in a thread for the stages of restarting that may be blocking, so as to not block the main thread.
create_analog_outputs(analog_properties)	
create_analog_widgets(channel_properties)	
create_dds_outputs(dds_properties)	
create_dds_widgets(channel_properties)	
create_digital_outputs(digital_properties)	
create_digital_widgets(channel_properties)	
create_image_outputs(image_properties)	
create_image_widgets(channel_properties)	
create_worker(name, WorkerClass[, workerargs])	Set up a worker process.
disconnect_restart_receiver(function)	
finalise_close_tab(currentpage)	
finalise_restart(currentpage)	
get_all_save_data()	
get_builtin_save_data()	Get builtin settings to be restored like whether the terminal is visible.
get_channel(channel)	
get_child_from_connection_table(...)	
get_front_panel_values()	
get_save_data()	
get_tab_layout()	
hide_error()	
initialise_GUI()	
initialise_workers()	
mainloop()	
on_force_full_buffered_reprogram()	

continues on next page

Table 3.2 – continued from previous page

on_resolve_value_inconsistency()	
program_device(*args, **kwargs)	
queue_work(worker_process, worker_function, ...)	
restart(*args)	
restore_builtin_save_data(data)	Restore builtin settings to be restored like whether the terminal is visible.
restore_save_data(data)	
set_tab_icon_and_colour()	Set the tab icon and the colour of its text to the values of self._tab_icon and self._tab_text_colour respectively
set_terminal_visible(visible)	
shutdown_workers(*args, **kwargs)	
start_run(*args, **kwargs)	
statemachine_timeout_add(delay, ...)	
statemachine_timeout_remove(statefunction)	
statemachine_timeout_remove_all()	
supports_remote_value_check(support)	
supports_smart_programming(support)	
transition_to_buffered(*args, **kwargs)	
transition_to_manual(*args, **kwargs)	
update_from_settings(settings)	

Attributes

ICON_BUSY
ICON_ERROR
ICON_FATAL_ERROR
ICON_OK
device_name
error_message
force_full_buffered_reprogram
mode
primary_worker
state

abort_buffered(**args*, ***kwargs*)

abort_transition_to_buffered(**args*, ***kwargs*)

add_secondary_worker(*worker*)

auto_create_widgets()

auto_place_widgets(**args*)

check_remote_values(**args*, ***kwargs*)

create_analog_outputs(*analog_properties*)

create_analog_widgets(*channel_properties*)

create_dds_outputs(*dds_properties*)

create_dds_widgets(*channel_properties*)

create_digital_outputs(*digital_properties*)

create_digital_widgets(*channel_properties*)

create_image_outputs(*image_properties*)

create_image_widgets(*channel_properties*)

get_channel(*channel*)

get_child_from_connection_table(*parent_device_name*, *port*)

get_front_panel_values()

get_save_data()

initialise_GUI()

initialise_workers()

on_resolve_value_inconsistency()

property primary_worker

program_device(**args*, ***kwargs*)

restore_save_data(*data*)

start_run(**args*, ***kwargs*)

supports_remote_value_check(*support*)

transition_to_buffered(**args*, ***kwargs*)

transition_to_manual(**args*, ***kwargs*)

update_from_settings(*settings*)

3.3.2 blacs.device_base_class.DeviceWorker

class blacs.device_base_class.**DeviceWorker**(**args*, ***kwargs*)

Bases: *Worker*

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
abort_buffered()	
abort_transition_to_buffered()	
check_remote_values()	
init()	
initialise()	
interrupt_startup([reason])	Called from the parent process.
mainloop()	
program_manual(front_panel_values)	
run(worker_name, device_name, extraargs)	The method that gets called in the subprocess.
shutdown()	
start(*args, **kwargs)	Call in the parent process to start a subprocess.
terminate([wait_timeout])	Interrupt process startup if not already done, ensuring self.child exists or is None if startup was interrupted before the process was created.
transition_to_buffered(device_name, h5file, ...)	
transition_to_manual()	

abort_buffered()

abort_transition_to_buffered()

check_remote_values()

init()

initialise()

program_manual(*front_panel_values*)

shutdown()

transition_to_buffered(*device_name*, *h5file*, *front_panel_values*, *refresh*)

transition_to_manual()

3.4 blacs.experiment_queue

Functions

tempfilename([prefix, suffix])	Return a filepath appropriate for use as a temporary file

3.4.1 blacs.experiment_queue.tempfilename

blacs.experiment_queue.**tempfilename**(*prefix='BLACS-temp-'*, *suffix='.h5'*)

Return a filepath appropriate for use as a temporary file

Classes

QueueManager(BLACS, ui)	
QueueTreeview(*args, **kwargs)	

3.4.2 blacs.experiment_queue.QueueManager

class blacs.experiment_queue.**QueueManager**(*BLACS*, *ui*)

Bases: object

__init__(*BLACS*, *ui*)

Methods

__init__(BLACS, ui)
append(h5files)
clean_h5_file(h5file, new_h5_file[, ...])
get_device_error_state(name, device_list)
get_next_file()
get_save_data()
get_status()
is_in_queue(path)
manage()
new_rep_name(h5_filepath)
on_add_shots_triggered()
prepend(h5file)
process_request(h5_filepath)
restore_save_data(data)
set_status(queue_status[, shot_filepath])
transition_device_to_buffered(name, ...)

Attributes

ICON_REPEAT
ICON_REPEAT_LAST
REPEAT_ALL
REPEAT_LAST
manager_paused
manager_repeat
manager_repeat_mode
manager_running

ICON_REPEAT = ':qtutils/fugue/arrow-repeat'

ICON_REPEAT_LAST = ':qtutils/fugue/arrow-repeat-once'

REPEAT_ALL = 0

REPEAT_LAST = 1

append(*h5files*)

clean_h5_file(*h5file*, *new_h5_file*, *repeat_number=0*)

get_device_error_state(*name*, *device_list*)

get_next_file()

get_save_data()

get_status()

is_in_queue(*path*)

manage()

property manager_paused

property manager_repeat

property manager_repeat_mode

property manager_running

new_rep_name(*h5_filepath*)

on_add_shots_triggered()

prepend(*h5file*)

process_request(*h5_filepath*)

restore_save_data(*data*)

set_status(*queue_status*, *shot_filepath=None*)

transition_device_to_buffered(*name*, *transition_list*, *h5file*, *restart_receiver*)

3.4.3 blacs.experiment_queue.QueueTreeview

class blacs.experiment_queue.**QueueTreeview**(**args*, ***kwargs*)

Bases: QTreeView

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)
acceptDrops(self)
accessibleDescription(self)
accessibleName(self)
actionEvent(self, a0)
actions(self)
activateWindow(self)
addAction(self, action)
addActions(self, actions)
addScrollBarWidget(self, widget, alignment)
adjustSize(self)
allColumnsShowFocus(self)
alternatingRowColors(self)
autoExpandDelay(self)
autoFillBackground(self)
autoScrollMargin(self)
backgroundRole(self)

continues on next page

Table 3.3 – continued from previous page

baseSize(self)
blockSignals(self, b)
changeEvent(self, a0)
childAt(-> Optional[QWidget])
childEvent(self, a0)
children(self)
childrenRect(self)
childrenRegion(self)
clearFocus(self)
clearMask(self)
clearSelection(self)
close(self)
closeEditor(self, editor, hint)
closeEvent(self, a0)
closePersistentEditor(self, index)
collapse(self, index)
collapseAll(self)
colorCount(self)
columnAt(self, x)
columnCountChanged(self, oldCount, newCount)
columnMoved(self)
columnResized(self, column, oldSize, newSize)
columnViewportPosition(self, column)
columnWidth(self, column)
commitData(self, editor)
connectNotify(self, signal)

continues on next page

Table 3.3 – continued from previous page

contentsMargins(self)
contentsRect(self)
contextMenuEvent(self, a0)
contextMenuPolicy(self)
cornerWidget(self)
create(self[, window, initializeWindow, ...])
createWindowContainer(window[, parent, flags])
currentChanged(self, current, previous)
currentIndex(self)
cursor(self)
customEvent(self, a0)
dataChanged(self, topLeft, bottomRight[, roles])
defaultDropAction(self)
deleteLater(self)
depth(self)
destroy(self[, destroyWindow, destroySubWindows])
devType(self)
devicePixelRatio(self)
devicePixelRatioF(self)
devicePixelRatioFScale()
dirtyRegionOffset(self)
disconnect(-> bool)
disconnectNotify(self, signal)
dragDropMode(self)
dragDropOverwriteMode(self)
dragEnabled(self)

continues on next page

Table 3.3 – continued from previous page

dragEnterEvent(self, e)
dragLeaveEvent(self, e)
dragMoveEvent(self, event)
drawBranches(self, painter, rect, index)
drawFrame(self, a0)
drawRow(self, painter, options, index)
drawTree(self, painter, region)
dropEvent(self, e)
dropIndicatorPosition(self)
dumpObjectInfo(self)
dumpObjectTree(self)
dynamicPropertyNames(self)
edit()
editTriggers(self)
editorDestroyed(self, editor)
effectiveWinId(self)
ensurePolished(self)
enterEvent(self, a0)
event(self, event)
eventFilter(self, object, event)
executeDelayedItemsLayout(self)
expand(self, index)
expandAll(self)
expandRecursively(self, index[, depth])
expandToDepth(self, depth)
expandsOnDoubleClick(self)

continues on next page

Table 3.3 – continued from previous page

find(a0)
findChild(-> QObjectT)
findChildren(...)
focusInEvent(self, e)
focusNextChild(self)
focusNextPrevChild(self, next)
focusOutEvent(self, e)
focusPolicy(self)
focusPreviousChild(self)
focusProxy(self)
focusWidget(self)
font(self)
fontInfo(self)
fontMetrics(self)
foregroundRole(self)
frameGeometry(self)
frameRect(self)
frameShadow(self)
frameShape(self)
frameSize(self)
frameStyle(self)
frameWidth(self)
geometry(self)
getContentsMargins(self)
grab(self[, rectangle])
grabGesture(self, type[, flags])

continues on next page

Table 3.3 – continued from previous page

grabKeyboard(self)
grabMouse()
grabShortcut(self, key[, context])
graphicsEffect(self)
graphicsProxyWidget(self)
hasAutoScroll(self)
hasFocus(self)
hasHeightForWidth(self)
hasMouseTracking(self)
hasTabletTracking(self)
header(self)
height(self)
heightForWidth(self, a0)
heightMM(self)
hide(self)
hideColumn(self, column)
hideEvent(self, a0)
horizontalOffset(self)
horizontalScrollBar(self)
horizontalScrollBarPolicy(self)
horizontalScrollMode(self)
horizontalScrollbarAction(self, action)
horizontalScrollbarValueChanged(self, value)
iconSize(self)
indentation(self)
indexAbove(self, index)

continues on next page

Table 3.3 – continued from previous page

indexAt(self, p)
indexBelow(self, index)
indexRowSizeHint(self, index)
indexWidget(self, index)
inherits(self, classname)
initPainter(self, painter)
initStyleOption(self, option)
inputMethodEvent(self, event)
inputMethodHints(self)
inputMethodQuery(self, query)
insertAction(self, before, action)
insertActions(self, before, actions)
installEventFilter(self, a0)
isActiveWindow(self)
isAncestorOf(self, child)
isAnimated(self)
isColumnHidden(self, column)
isEnabled(self)
isEnabledTo(self, a0)
isExpanded(self, index)
isFirstColumnSpanned(self, row, parent)
isFullScreen(self)
isHeaderHidden(self)
isHidden(self)
isIndexHidden(self, index)
isLeftToRight(self)

continues on next page

Table 3.3 – continued from previous page

isMaximized(self)	
isMinimized(self)	
isModal(self)	
isPersistentEditorOpen(self, index)	
isRightToLeft(self)	
isRowHidden(self, row, parent)	
isSignalConnected(self, signal)	
isSortingEnabled(self)	
isVisible(self)	
isVisibleTo(self, a0)	
isWidgetType(self)	
isWindow(self)	
isWindowModified(self)	
isWindowType(self)	
itemDelegate(-> tional[QAbstractItemDelegate])	Op-
itemDelegateForColumn(self, column)	
itemDelegateForRow(self, row)	
itemsExpandable(self)	
keyPressEvent(self, event)	
keyReleaseEvent(self, a0)	
keyboardGrabber()	
keyboardSearch(self, search)	
killTimer(self, id)	
layout(self)	
layoutDirection(self)	
leaveEvent(self, a0)	

continues on next page

Table 3.3 – continued from previous page

lineWidth(self)
locale(self)
logicalDpiX(self)
logicalDpiY(self)
lower(self)
mapFrom(self, a0, a1)
mapFromGlobal(self, a0)
mapFromParent(self, a0)
mapTo(self, a0, a1)
mapToGlobal(self, a0)
mapToParent(self, a0)
mask(self)
maximumHeight(self)
maximumSize(self)
maximumViewportSize(self)
maximumWidth(self)
metaObject(self)
metric(self, a0)
midLineWidth(self)
minimumHeight(self)
minimumSize(self)
minimumSizeHint(self)
minimumWidth(self)
model(self)
mouseDoubleClickEvent(self, e)
mouseGrabber()

continues on next page

Table 3.3 – continued from previous page

mouseMoveEvent(self, event)
mousePressEvent(self, e)
mouseReleaseEvent(self, event)
move()
moveCursor(self, cursorAction, modifiers)
moveEvent(self, a0)
moveToThread(self, thread)
nativeEvent(self, eventType, message)
nativeParentWidget(self)
nextInFocusChain(self)
normalGeometry(self)
objectName(self)
openPersistentEditor(self, index)
overrideWindowFlags(self, type)
overrideWindowState(self, state)
paintEngine(self)
paintEvent(self, e)
paintingActive(self)
palette(self)
parent(self)
parentWidget(self)
physicalDpiX(self)
physicalDpiY(self)
pos(self)
previousInFocusChain(self)
property(self, name)

continues on next page

Table 3.3 – continued from previous page

pyqtConfigure(...)	Each keyword argument is either the name of a Qt property or a Qt signal.
raise_(self)	
receivers(self, signal)	
rect(self)	
reexpand(self)	
releaseKeyboard(self)	
releaseMouse(self)	
releaseShortcut(self, id)	
removeAction(self, action)	
removeEventFilter(self, a0)	
render(, sourceRegion, flags, ...)	
repaint(-> None -> None)	
reset(self)	
resetHorizontalScrollMode(self)	
resetIndentation(self)	
resetVerticalScrollMode(self)	
resize()	
resizeColumnToContents(self, column)	
resizeEvent(self, e)	
restoreGeometry(self, geometry)	
rootIndex(self)	
rootIsDecorated(self)	
rowHeight(self, index)	
rowsAboutToBeRemoved(self, parent, start, end)	
rowsInserted(self, parent, start, end)	
rowsRemoved(self, parent, first, last)	

continues on next page

Table 3.3 – continued from previous page

saveGeometry(self)
scheduleDelayedItemsLayout(self)
screen(self)
scroll()
scrollBarWidgets(self, alignment)
scrollContentsBy(self, dx, dy)
scrollDirtyRegion(self, dx, dy)
scrollTo(self, index[, hint])
scrollToBottom(self)
scrollToTop(self)
selectAll(self)
selectedIndexes(self)
selectionBehavior(self)
selectionChanged(self, selected, deselected)
selectionCommand(self, index[, event])
selectionMode(self)
selectionModel(self)
sender(self)
senderSignalIndex(self)
setAcceptDrops(self, on)
setAccessibleDescription(self, description)
setAccessibleName(self, name)
setAllColumnsShowFocus(self, enable)
setAlternatingRowColors(self, enable)
setAnimated(self, enable)
setAttribute(self, attribute[, on])

continues on next page

Table 3.3 – continued from previous page

setAutoExpandDelay(self, delay)
setAutoFillBackground(self, enabled)
setAutoScroll(self, enable)
setAutoScrollMargin(self, margin)
setBackgroundRole(self, a0)
setBaseSize()
setColumnHidden(self, column, hide)
setColumnWidth(self, column, width)
setContentsMargins()
setContextMenuPolicy(self, policy)
setCornerWidget(self, widget)
setCurrentIndex(self, index)
setCursor(self, a0)
setDefaultDropAction(self, dropAction)
setDirtyRegion(self, region)
setDisabled(self, a0)
setDragDropMode(self, behavior)
setDragDropOverwriteMode(self, overwrite)
setDragEnabled(self, enable)
setDropIndicatorShown(self, enable)
setEditTriggers(self, triggers)
setEnabled(self, a0)
setExpanded(self, index, expand)
setExpandsOnDoubleClick(self, enable)
setFirstColumnSpanned(self, row, parent, span)
setFixedHeight(self, h)

continues on next page

Table 3.3 – continued from previous page

setFixedSize()
setFixedWidth(self, w)
setFocus()
setFocusPolicy(self, policy)
setFocusProxy(self, a0)
setFont(self, a0)
setForegroundRole(self, a0)
setFrameRect(self, a0)
setFrameShadow(self, a0)
setFrameShape(self, a0)
setFrameStyle(self, a0)
setGeometry()
setGraphicsEffect(self, effect)
setHeader(self, header)
setHeaderHidden(self, hide)
setHidden(self, hidden)
setHorizontalScrollBar(self, scrollbar)
setHorizontalScrollBarPolicy(self, a0)
setHorizontalScrollMode(self, mode)
setIconSize(self, size)
setIndentation(self, i)
setIndexWidget(self, index, widget)
setInputMethodHints(self, hints)
setItemDelegate(self, delegate)
setItemDelegateForColumn(self, column, delegate)
setItemDelegateForRow(self, row, delegate)

continues on next page

Table 3.3 – continued from previous page

setItemsExpandable(self, enable)
setLayout(self, a0)
setLayoutDirection(self, direction)
setLineWidth(self, a0)
setLocale(self, locale)
setMask()
setMaximumHeight(self, maxh)
setMaximumSize()
setMaximumWidth(self, maxw)
setMidLineWidth(self, a0)
setMinimumHeight(self, minh)
setMinimumSize()
setMinimumWidth(self, minw)
setModel(self, model)
setMouseTracking(self, enable)
setObjectName(self, name)
setPalette(self, a0)
setParent()
setProperty(self, name, value)
setRootIndex(self, index)
setRootIsDecorated(self, show)
setRowHidden(self, row, parent, hide)
setSelection(self, rect, command)
setSelectionBehavior(self, behavior)
setSelectionMode(self, mode)
setSelectionModel(self, selectionModel)

continues on next page

Table 3.3 – continued from previous page

setShortcutAutoRepeat(self, id[, enabled])
setShortcutEnabled(self, id[, enabled])
setSizeAdjustPolicy(self, policy)
setSizeIncrement()
setSizePolicy()
setSortingEnabled(self, enable)
setState(self, state)
setStatusTip(self, a0)
setStyle(self, a0)
setStyleSheet(self, styleSheet)
setTabKeyNavigation(self, enable)
setTabOrder(a0, a1)
setTabletTracking(self, enable)
setTextElideMode(self, mode)
setToolTip(self, a0)
setToolTipDuration(self, msec)
setTreePosition(self, logicalIndex)
setUniformRowHeights(self, uniform)
setUpdatesEnabled(self, enable)
setVerticalScrollBar(self, scrollbar)
setVerticalScrollBarPolicy(self, a0)
setVerticalScrollMode(self, mode)
setViewport(self, widget)
setViewportMargins()
setVisible(self, visible)
setWhatsThis(self, a0)

continues on next page

Table 3.3 – continued from previous page

setWindowFilePath(self, filePath)
setWindowFlag(self, a0[, on])
setWindowFlags(self, type)
setWindowIcon(self, icon)
setWindowIconText(self, a0)
setWindowModality(self, windowModality)
setWindowModified(self, a0)
setWindowOpacity(self, level)
setWindowRole(self, a0)
setWindowState(self, state)
setWindowTitle(self, a0)
setWordWrap(self, on)
setupViewport(self, viewport)
sharedPainter(self)
show(self)
showColumn(self, column)
showDropIndicator(self)
showEvent(self, a0)
showFullScreen(self)
showMaximized(self)
showMinimized(self)
showNormal(self)
signalsBlocked(self)
size(self)
sizeAdjustPolicy(self)
sizeHint(self)

continues on next page

Table 3.3 – continued from previous page

sizeHintForColumn(self, column)
sizeHintForIndex(self, index)
sizeHintForRow(self, row)
sizeIncrement(self)
sizePolicy(self)
sortByColumn(self, column, order)
stackUnder(self, a0)
startDrag(self, supportedActions)
startTimer(self, interval[, timerType])
state(self)
statusTip(self)
style(self)
styleSheet(self)
tabKeyNavigation(self)
tabletEvent(self, a0)
testAttribute(self, attribute)
textElideMode(self)
thread(self)
timerEvent(self, event)
toolTip(self)
toolTipDuration(self)
tr(self, sourceText[, disambiguation, n])
treePosition(self)
underMouse(self)
ungrabGesture(self, type)
uniformRowHeights(self)

continues on next page

Table 3.3 – continued from previous page

unsetCursor(self)
unsetLayoutDirection(self)
unsetLocale(self)
update()
updateEditorData(self)
updateEditorGeometries(self)
updateGeometries(self)
updateGeometry(self)
updateMicroFocus(self)
updatesEnabled(self)
verticalOffset(self)
verticalScrollBar(self)
verticalScrollBarPolicy(self)
verticalScrollMode(self)
verticalScrollbarAction(self, action)
verticalScrollbarValueChanged(self, value)
viewOptions(self)
viewport(self)
viewportEvent(self, event)
viewportMargins(self)
viewportSizeHint(self)
visibleRegion(self)
visualRect(self, index)
visualRegionForSelection(self, selection)
whatsThis(self)
wheelEvent(self, a0)

continues on next page

Table 3.3 – continued from previous page

width(self)
widthMM(self)
winId(self)
window(self)
windowFilePath(self)
windowFlags(self)
windowHandle(self)
windowIcon(self)
windowIconText(self)
windowModality(self)
windowOpacity(self)
windowRole(self)
windowState(self)
windowTitle(self)
windowType(self)
wordWrap(self)
x(self)
y(self)

Attributes

AboveItem
AdjustIgnored
AdjustToContents
AdjustToContentsOnFirstShow
AllEditTriggers

continues on next page

Table 3.4 – continued from previous page

AnimatingState
AnyKeyPressed
BelowItem
Box
CollapsingState
ContiguousSelection
CurrentChanged
DoubleClicked
DragDrop
DragOnly
DragSelectingState
DraggingState
DrawChildren
DrawWindowBackground
DropOnly
EditKeyPressed
EditingState
EnsureVisible
ExpandingState
ExtendedSelection
HLine
IgnoreMask
InternalMove
MoveDown
MoveEnd
MoveHome

continues on next page

Table 3.4 – continued from previous page

MoveLeft
MoveNext
MovePageDown
MovePageUp
MovePrevious
MoveRight
MoveUp
MultiSelection
NoDragDrop
NoEditTriggers
NoFrame
NoSelection
NoState
OnItem
OnViewport
Panel
PdmDepth
PdmDevicePixelRatio
PdmDevicePixelRatioScaled
PdmDpiX
PdmDpiY
PdmHeight
PdmHeightMM
PdmNumColors
PdmPhysicalDpiX
PdmPhysicalDpiY

continues on next page

Table 3.4 – continued from previous page

PdmWidth	
PdmWidthMM	
Plain	
PositionAtBottom	
PositionAtCenter	
PositionAtTop	
Raised	
ScrollPerItem	
ScrollPerPixel	
SelectColumns	
SelectItems	
SelectRows	
SelectedClicked	
Shadow_Mask	
Shape_Mask	
SingleSelection	
StyledPanel	
Sunken	
VLine	
WinPanel	
activated	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
clicked	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
collapsed	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
customContextMenuRequested	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
destroyed	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
doubleClicked	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL

continues on next page

Table 3.4 – continued from previous page

entered	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL	
expanded	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL	
iconSizeChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL	
objectNameChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL	
pressed	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL	
staticMetaObject		
viewportEntered	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL	
windowIconChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL	
windowIconTextChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL	
windowTitleChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL	

dragEnterEvent(*self*, *e: QDragEnterEvent | None*)

dragMoveEvent(*self*, *event: QDragMoveEvent | None*)

dropEvent(*self*, *e: QDropEvent | None*)

keyPressEvent(*self*, *event: QKeyEvent | None*)

3.5 blacs.front_panel_settings

Classes

FrontPanelSettings(settings_path, ...)

3.5.1 blacs.front_panel_settings.FrontPanelSettings

class blacs.front_panel_settings.**FrontPanelSettings**(*settings_path*, *connection_table*)

Bases: object

__init__(*settings_path*, *connection_table*)

Methods

__init__(settings_path, connection_table)	
check_row(row, ct_match, blacs_ct, saved_ct)	
get_save_data()	
handle_return_code(row, result, settings, ...)	
restore()	
save_front_panel_to_h5(current_file, states, ...)	
setup(blacs)	
store_front_panel_in_h5(hdf5_file, tab_data, ...)	

check_row(*row*, *ct_match*, *blacs_ct*, *saved_ct*)

get_save_data()

handle_return_code(*row*, *result*, *settings*, *question*, *error*)

restore()

save_front_panel_to_h5(*current_file*, *states*, *tab_positions*, *window_data*, *plugin_data*, *silent={}*, *force_new_conn_table=False*)

setup(*blacs*)

store_front_panel_in_h5(*hdf5_file*, *tab_data*, *notebook_data*, *window_data*, *plugin_data*, *save_conn_table=False*, *save_queue_data=True*)

3.6 blacs.notifications

Classes

Notifications(BLACS)	

3.6.1 blacs.notifications.Notifications

class blacs.notifications.**Notifications**(*BLACS*)

Bases: object

__init__(*BLACS*)

Methods

__init__(BLACS)
add_notification(notification_class)
close_all()
close_notification(notification_class, callback)
get_instance(notification_class)
get_state(notification_class)
minimize_notification(notification_class, ...)
show_notification(notification_class, callback)

add_notification(*notification_class*)

close_all()

close_notification(*notification_class*, *callback*)

get_instance(*notification_class*)

get_state(*notification_class*)

minimize_notification(*notification_class*, *callback*)

show_notification(*notification_class*, *callback*)

3.7 blacs.output_classes

Classes

AO(hardware_name, connection_name, ...)	
DDS(hardware_name, connection_name, output_list)	
DO(hardware_name, connection_name, ...)	
Image(hardware_name, connection_name, ...[, ...])	

3.7.1 blacs.output_classes.AO

class blacs.output_classes.**AO**(*hardware_name*, *connection_name*, *device_name*, *program_function*, *settings*, *calib_class*, *calib_params*, *default_units*, *min*, *max*, *step*, *decimals*)

Bases: object

__init__(*hardware_name*, *connection_name*, *device_name*, *program_function*, *settings*, *calib_class*, *calib_params*, *default_units*, *min*, *max*, *step*, *decimals*)

Methods

__init__(hardware_name, connection_name, ...)	
add_widget(widget)	
change_unit(unit[, program])	
convert_range_from_base(value, range, unit)	
convert_range_to_base(value, range, unit)	
convert_value_from_base(value, unit)	
convert_value_to_base(value, unit)	
create_widget([display_name, ...])	
get_step_size(unit)	
lock()	
remove_widget(widget[, call_set_AO, new_AO])	
set_step_size(step_size, unit)	
set_value(value[, unit, program])	
unlock()	

Attributes

name
value

add_widget(*widget*)

change_unit(*unit*, *program=True*)

convert_range_from_base(*value*, *range*, *unit*)

convert_range_to_base(*value*, *range*, *unit*)

convert_value_from_base(*value*, *unit*)

convert_value_to_base(*value*, *unit*)

create_widget(*display_name=None*, *horizontal_alignment=False*, *parent=None*)

get_step_size(*unit*)

lock()

property name

remove_widget(*widget*, *call_set_AO=True*, *new_AO=None*)

set_step_size(*step_size*, *unit*)

set_value(*value*, *unit=None*, *program=True*)

unlock()

property value

3.7.2 blacs.output_classes.DDS

class blacs.output_classes.**DDS**(*hardware_name*, *connection_name*, *output_list*)

Bases: object

__init__(*hardware_name*, *connection_name*, *output_list*)

Methods

__init__(hardware_name, connection_name, ...)
add_widget(widget)
create_widget(*args, **kwargs)
get_subchnl_list()
get_unused_subchnl_list()
remove_widget(widget)
set_value(value[, program])

Attributes

name
value

add_widget(*widget*)

create_widget(**args*, ***kwargs*)

get_subchnl_list()

get_unused_subchnl_list()

property name

remove_widget(*widget*)

set_value(*value*, *program=True*)

property value

3.7.3 blacs.output_classes.DO

class blacs.output_classes.**DO**(*hardware_name*, *connection_name*, *device_name*, *program_function*, *settings*)

Bases: object

__init__(*hardware_name*, *connection_name*, *device_name*, *program_function*, *settings*)

Methods

__init__(hardware_name, connection_name, ...)	
add_widget(widget[, inverted])	
create_widget(*args, **kwargs)	
lock()	
remove_widget(widget)	
set_value(state[, program])	
unlock()	

Attributes

name	
value	

add_widget(*widget*, *inverted=False*)

create_widget(**args*, ***kwargs*)

lock()

property name

remove_widget(*widget*)

set_value(*state*, *program=True*)

unlock()

property value

3.7.4 blacs.output_classes.Image

class blacs.output_classes.**Image**(*hardware_name*, *connection_name*, *device_name*, *program_function*, *settings*, *width*, *height*, *x=None*, *y=None*)

Bases: object

__init__(*hardware_name*, *connection_name*, *device_name*, *program_function*, *settings*, *width*, *height*, *x=None*, *y=None*)

Methods

__init__(hardware_name, connection_name, ...)	
add_widget(widget)	
create_widget(*args, **kwargs)	
lock()	
remove_widget(widget)	
set_value(value[, program])	
unlock()	

Attributes

name	
value	

add_widget(*widget*)

create_widget(**args*, ***kwargs*)

lock()

property name

remove_widget(*widget*)

set_value(*value*, *program=True*)

unlock()

property value

3.8 blacs.plugins

Functions

get_callbacks(name)	Return all the callbacks for a particular name, in order of priority

3.8.1 blacs.plugins.get_callbacks

blacs.plugins.**get_callbacks**(*name*)

Return all the callbacks for a particular name, in order of priority

Classes

Callback(func[, priority])	Class wrapping a callable.
callback([priority])	Decorator to turn a function into a Callback object.

3.8.2 blacs.plugins.Callback

class blacs.plugins.**Callback**(*func*, *priority=10*)

Bases: object

Class wrapping a callable. At present only differs from a regular function in that it has a "priority" attribute - lower numbers means higher priority. If there are multiple callbacks triggered by the same event, they will be returned in order of priority by get_callbacks

__init__(*func*, *priority=10*)

Methods

__init__(func[, priority])

3.8.3 blacs.plugins.callback

class blacs.plugins.**callback**(*priority=10*)

Bases: object

Decorator to turn a function into a Callback object. Presently optional, and only required if the callback needs to have a non-default priority set

__init__(*priority=10*)

Methods

__init__([priority])

Modules

blacs.plugins.connection_table
blacs.plugins.cycle_time
blacs.plugins.delete_repeated_shots
blacs.plugins.general
blacs.plugins.memory
blacs.plugins.progress_bar
blacs.plugins.theme

3.8.4 blacs.plugins.connection_table

Classes

BrokenDevicesNotification(BLACS)
Menu(BLACS)
Plugin(initial_settings)
RecompileNotification(BLACS)
Setting(data)

blacs.plugins.connection_table.BrokenDevicesNotification

class blacs.plugins.connection_table.**BrokenDevicesNotification**(*BLACS*)

Bases: object

__init__(*BLACS*)

Methods

__init__(BLACS)	
close()	
get_properties()	
get_save_data()	
get_widget()	
set_broken_devices(device_names)	
set_functions(show_func, hide_func, ...)	

Attributes

name	

close()

get_properties()

get_save_data()

get_widget()

name = 'Device initialization failed'

set_broken_devices(*device_names*)

set_functions(*show_func*, *hide_func*, *close_func*, *get_state*)

blacs.plugins.connection_table.Menu

class blacs.plugins.connection_table.**Menu**(*BLACS*)

Bases: object

__init__(*BLACS*)

Methods

__init__(BLACS)	
get_menu_items()	
on_edit_connection_table(*args, **kwargs)	
on_recompile_connection_table(*args, **kwargs)	
on_select_globals(*args, **kwargs)	

get_menu_items()

on_edit_connection_table(**args*, ***kwargs*)

on_recompile_connection_table(**args*, ***kwargs*)

on_select_globals(**args*, ***kwargs*)

blacs.plugins.connection_table.Plugin

class blacs.plugins.connection_table.**Plugin**(*initial_settings*)

Bases: object

__init__(*initial_settings*)

Methods

__init__(initial_settings)	
close()	
get_callbacks()	
get_menu_class()	
get_notification_classes()	
get_save_data()	
get_setting_classes()	
plugin_setup_complete(BLACS)	
set_menu_instance(menu)	
set_notification_instances(notifications)	

close()

get_callbacks()

get_menu_class()

get_notification_classes()

get_save_data()

get_setting_classes()

plugin_setup_complete(*BLACS*)

set_menu_instance(*menu*)

set_notification_instances(*notifications*)

blacs.plugins.connection_table.RecompileNotification

class blacs.plugins.connection_table.**RecompileNotification**(*BLACS*)

Bases: object

__init__(*BLACS*)

Methods

__init__(BLACS)	
callback(name, info[, event])	
close()	
get_properties()	
get_save_data()	
get_widget()	
on_recompile_connection_table(*args, **kwargs)	
on_restart()	
set_functions(show_func, hide_func, ...)	
setup_filewatching([clean_modified_info])	

Attributes

name

callback(*name*, *info*, *event=None*)

close()

get_properties()

get_save_data()

get_widget()

name = 'Connection Table'

on_recompile_connection_table(**args*, ***kwargs*)

on_restart()

set_functions(*show_func*, *hide_func*, *close_func*, *get_state*)

setup_filewatching(*clean_modified_info=None*)

blacs.plugins.connection_table.Setting

class blacs.plugins.connection_table.**Setting**(*data*)

Bases: object

__init__(*data*)

Methods

__init__(data)
add_calibration_file()
add_calibration_folder()
add_global_file(*args, **kwargs)
calibrations_sort_indicator_changed()
close()
create_dialog(notebook)
delete_selected_conversion_file()
delete_selected_globals_file()
enum_to_order(enum)
get_value(name)
global_sort_indicator_changed()
is_filepath_in_store(filepath, store)
order_to_enum(order)
save()

Attributes

name

add_calibration_file()

add_calibration_folder()

add_global_file(**args*, ***kwargs*)

calibrations_sort_indicator_changed()

close()

create_dialog(*notebook*)

delete_selected_conversion_file()

delete_selected_globals_file()

enum_to_order(*enum*)

get_value(*name*)

global_sort_indicator_changed()

is_filepath_in_store(*filepath*, *store*)

name = 'Connection Table'

order_to_enum(*order*)

save()

3.8.5 blacs.plugins.cycle_time

Classes

Plugin(initial_settings)

blacs.plugins.cycle_time.Plugin

class blacs.plugins.cycle_time.**Plugin**(*initial_settings*)

Bases: object

__init__(*initial_settings*)

Methods

__init__(initial_settings)	
close()	
do_delay(h5_filepath)	
get_callbacks()	
get_menu_class()	
get_notification_classes()	
get_save_data()	
get_setting_classes()	
plugin_setup_complete(BLACS)	
set_menu_instance(menu)	
set_notification_instances(notifications)	

Attributes

pre_transition_to_buffered	Class wrapping a callable.
science_starting	Class wrapping a callable.

close()

do_delay(*h5_filepath*)

get_callbacks()

get_menu_class()

get_notification_classes()

get_save_data()

get_setting_classes()

plugin_setup_complete(*BLACS*)

pre_transition_to_buffered

Class wrapping a callable. At present only differs from a regular function in that it has a "priority" attribute - lower numbers means higher priority. If there are multiple callbacks triggered by the same event, they will be returned in order of priority by get_callbacks

science_starting

Class wrapping a callable. At present only differs from a regular function in that it has a "priority" attribute - lower numbers means higher priority. If there are multiple callbacks triggered by the same event, they will be returned in order of priority by get_callbacks

set_menu_instance(*menu*)

set_notification_instances(*notifications*)

3.8.6 blacs.plugins.delete_repeated_shots

Classes

Plugin(initial_settings)	

blacs.plugins.delete_repeated_shots.Plugin

class blacs.plugins.delete_repeated_shots.**Plugin**(*initial_settings*)

Bases: object

__init__(*initial_settings*)

Methods

__init__(initial_settings)
close()
get_callbacks()
get_menu_class()
get_notification_classes()
get_save_data()
get_setting_classes()
mainloop()
on_reset_button_clicked()
on_shot_complete(h5_filepath)
on_spinbox_value_changed(value)
plugin_setup_complete(BLACS)
set_menu_instance(menu)
set_notification_instances(notifications)

close()

get_callbacks()

get_menu_class()

get_notification_classes()

get_save_data()

get_setting_classes()

mainloop()

on_reset_button_clicked()

on_shot_complete(*h5_filepath*)

on_spinbox_value_changed(*value*)

plugin_setup_complete(*BLACS*)

set_menu_instance(*menu*)

set_notification_instances(*notifications*)

3.8.7 blacs.plugins.general

Classes

Plugin(initial_settings)	
Setting(data)	

blacs.plugins.general.Plugin

class blacs.plugins.general.**Plugin**(*initial_settings*)

Bases: object

__init__(*initial_settings*)

Methods

__init__(initial_settings)	
close()	
get_callbacks()	
get_menu_class()	
get_notification_classes()	
get_save_data()	
get_setting_classes()	
plugin_setup_complete(BLACS)	
set_menu_instance(menu)	
set_notification_instances(notifications)	

close()

get_callbacks()

get_menu_class()

get_notification_classes()

get_save_data()

get_setting_classes()

plugin_setup_complete(*BLACS*)

set_menu_instance(*menu*)

set_notification_instances(*notifications*)

blacs.plugins.general.Setting

class blacs.plugins.general.**Setting**(*data*)

Bases: object

__init__(*data*)

Methods

__init__(data)
close()
create_dialog(notebook)
get_value(name)
save()

Attributes

name

close()

create_dialog(*notebook*)

get_value(*name*)

name = 'General'

save()

3.8.8 blacs.plugins.memory

Classes

Menu(BLACS)	
Plugin(initial_settings)	

blacs.plugins.memory.Menu

class blacs.plugins.memory.**Menu**(*BLACS*)

Bases: object

__init__(*BLACS*)

Methods

__init__(BLACS)	
get_menu_items()	

get_menu_items()

blacs.plugins.memory.Plugin

class blacs.plugins.memory.**Plugin**(*initial_settings*)

Bases: object

__init__(*initial_settings*)

Methods

__init__(initial_settings)	
close()	
get_callbacks()	
get_menu_class()	
get_notification_classes()	
get_save_data()	
get_setting_classes()	
plugin_setup_complete(BLACS)	
set_menu_instance(menu)	
set_notification_instances(notifications)	

close()

get_callbacks()

get_menu_class()

get_notification_classes()

get_save_data()

get_setting_classes()

plugin_setup_complete(*BLACS*)

set_menu_instance(*menu*)

set_notification_instances(*notifications*)

3.8.9 blacs.plugins.progress_bar

Functions

black_has_good_contrast(r, g, b)	Return whether black text or white text would have better contrast on a background of the given colour, according to W3C recommendations (see https://www.w3.org/TR/WCAG20/).

blacs.plugins.progress_bar.black_has_good_contrast

blacs.plugins.progress_bar.**black_has_good_contrast**(*r*, *g*, *b*)

Return whether black text or white text would have better contrast on a background of the given colour, according to W3C recommendations (see https://www.w3.org/TR/WCAG20/). Return True for black or False for white

Classes

Plugin(initial_settings)	

blacs.plugins.progress_bar.Plugin

class blacs.plugins.progress_bar.**Plugin**(*initial_settings*)

Bases: object

__init__(*initial_settings*)

Methods

__init__(initial_settings)	
clear_bar()	
close()	
get_callbacks()	
get_menu_class()	
get_next_thing()	Figure out what's going to happen next: a wait, a time marker, or a regular update.
get_notification_classes()	
get_save_data()	
get_setting_classes()	
mainloop()	
plugin_setup_complete(BLACS)	
set_menu_instance(menu)	
set_notification_instances(notifications)	
update_bar_style([marker, wait, previous])	Update the bar's style to reflect the next marker or wait, according to self.next_marker_index or self.next_wait_index.
update_bar_value([marker, wait])	Update the progress bar with the current time elapsed.

Attributes

on_science_over	Class wrapping a callable.
on_science_starting	Class wrapping a callable.

clear_bar()

close()

get_callbacks()

get_menu_class()

get_next_thing()

Figure out what's going to happen next: a wait, a time marker, or a regular update. Return a string saying which, and a float saying how long from now it will occur. If the thing has already happened but not been taken into account by our processing yet, then return zero for the time.

get_notification_classes()

get_save_data()

get_setting_classes()

mainloop()

on_science_over

Class wrapping a callable. At present only differs from a regular function in that it has a "priority" attribute - lower numbers means higher priority. If there are multiple callbacks triggered by the same event, they will be returned in order of priority by get_callbacks

on_science_starting

Class wrapping a callable. At present only differs from a regular function in that it has a "priority" attribute - lower numbers means higher priority. If there are multiple callbacks triggered by the same event, they will be returned in order of priority by get_callbacks

plugin_setup_complete(*BLACS*)

set_menu_instance(*menu*)

set_notification_instances(*notifications*)

update_bar_style(*marker=False*, *wait=False*, *previous=False*)

Update the bar's style to reflect the next marker or wait, according to self.next_marker_index or self.next_wait_index. If previous=True, instead update to reflect the current marker or wait.

update_bar_value(*marker=False*, *wait=False*)

Update the progress bar with the current time elapsed. If marker or wait is true, then use the exact time at which the next marker or wait is defined, rather than the current time as returned by time.time()

3.8.10 blacs.plugins.theme

Functions

is_default_stylesheet(stylesheet)	Return whether a stylesheet is the same as the default stylesheet, modulo whitespace

blacs.plugins.theme.is_default_stylesheet

blacs.plugins.theme.**is_default_stylesheet**(*stylesheet*)

Return whether a stylesheet is the same as the default stylesheet, modulo whitespace

Classes

Plugin(initial_settings)	
Setting(data)	

blacs.plugins.theme.Plugin

class blacs.plugins.theme.**Plugin**(*initial_settings*)

Bases: object

__init__(*initial_settings*)

Methods

__init__(initial_settings)	
close()	
get_callbacks()	
get_menu_class()	
get_notification_classes()	
get_save_data()	
get_setting_classes()	
plugin_setup_complete(BLACS)	
set_menu_instance(menu)	
set_notification_instances(notifications)	
update_stylesheet()	

close()

get_callbacks()

get_menu_class()

get_notification_classes()

get_save_data()

get_setting_classes()

plugin_setup_complete(*BLACS*)

set_menu_instance(*menu*)

set_notification_instances(*notifications*)

update_stylesheet()

blacs.plugins.theme.Setting

class blacs.plugins.theme.**Setting**(*data*)

Bases: object

__init__(*data*)

Methods

__init__(data)
close()
create_dialog(notebook)
get_value(name)
on_set_green_button_theme()
save()

Attributes

name

close()

create_dialog(*notebook*)

get_value(*name*)

name = 'GUI Theme'

on_set_green_button_theme()

save()

3.9 blacs.tab_base_classes

Functions

define_state(allowed_modes, ...[, ...])	
get_unique_id()	

3.9.1 blacs.tab_base_classes.define_state

blacs.tab_base_classes.**define_state**(*allowed_modes*, *queue_state_indefinitely*, *delete_stale_states=False*)

3.9.2 blacs.tab_base_classes.get_unique_id

blacs.tab_base_classes.**get_unique_id**()

Classes

Counter()	A class with a single method that returns a different integer each time it's called.
MyTab(notebook, settings[, restart])	
MyWorker(*args, **kwargs)	
PluginTab(notebook, settings)	
StateQueue(device_name)	
Tab(notebook, settings[, restart])	
Worker(*args, **kwargs)	

3.9.3 blacs.tab_base_classes.Counter

class blacs.tab_base_classes.**Counter**

Bases: object

A class with a single method that returns a different integer each time it's called.

__init__()

Methods

__init__()
get()

get()

3.9.4 blacs.tab_base_classes.MyTab

class blacs.tab_base_classes.**MyTab**(*notebook*, *settings*, *restart=False*)

Bases: *Tab*

__init__(*notebook*, *settings*, *restart=False*)

Methods

__init__(notebook, settings[, restart])	
add_baz_timeout()	
bar(*args, **kwargs)	
baz(*args, **kwargs)	
baz_unpickleable(*args, **kwargs)	
check_time()	
clean_ui_on_restart()	
close_tab([finalise])	Close the tab, terminate subprocesses and join the mainloop thread.
connect_restart_receiver(function)	
continue_restart(currentpage)	Called in a thread for the stages of restarting that may be blocking, so as to not block the main thread.
create_worker(name, WorkerClass[, workerargs])	Set up a worker process.
disconnect_restart_receiver(function)	
fatal()	
finalise_close_tab(currentpage)	
finalise_restart(currentpage)	
foo(*args, **kwargs)	

continues on next page

Table 3.5 – continued from previous page

get_all_save_data()	
get_builtin_save_data()	Get builtin settings to be restored like whether the terminal is visible.
get_tab_layout()	
hide_error()	
initUI()	
mainloop()	
on_force_full_buffered_reprogram()	
queue_work(worker_process, worker_function, ...)	
remove_baz_timeout()	
restart(*args)	
restore_builtin_save_data(data)	Restore builtin settings to be restored like whether the terminal is visible.
set_tab_icon_and_colour()	Set the tab icon and the colour of its text to the values of self._tab_icon and self._tab_text_colour respectively
set_terminal_visible(visible)	
shutdown_workers(*args, **kwargs)	
statemachine_timeout_add(delay, ...)	
statemachine_timeout_remove(statefunction)	
statemachine_timeout_remove_all()	
supports_smart_programming(support)	
update_from_settings(settings)	

Attributes

ICON_BUSY
ICON_ERROR
ICON_FATAL_ERROR
ICON_OK
device_name
error_message
force_full_buffered_reprogram
mode
state

add_baz_timeout()

bar(**args*, ***kwargs*)

baz(**args*, ***kwargs*)

baz_unpickleable(**args*, ***kwargs*)

fatal()

foo(**args*, ***kwargs*)

initUI()

remove_baz_timeout()

3.9.5 blacs.tab_base_classes.MyWorker

class blacs.tab_base_classes.**MyWorker**(**args*, ***kwargs*)

Bases: *Worker*

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
bar(*args, **kwargs)	
baz(zzz, *args, **kwargs)	
foo(*args, **kwargs)	
init()	
interrupt_startup([reason])	Called from the parent process.
mainloop()	
run(worker_name, device_name, extraargs)	The method that gets called in the subprocess.
start(*args, **kwargs)	Call in the parent process to start a subprocess.
terminate([wait_timeout])	Interrupt process startup if not already done, ensuring self.child exists or is None if startup was interrupted before the process was created.

bar(**args*, ***kwargs*)

baz(*zzz*, **args*, ***kwargs*)

foo(**args*, ***kwargs*)

init()

3.9.6 blacs.tab_base_classes.PluginTab

class blacs.tab_base_classes.**PluginTab**(*notebook*, *settings*)

Bases: object

__init__(*notebook*, *settings*)

Methods

__init__(notebook, settings)	
close_tab(**kwargs)	
get_all_save_data()	
get_builtin_save_data()	
get_save_data()	
get_tab_layout()	
initialise_GUI()	
restore_save_data(data)	
set_tab_icon_and_colour()	Set the tab icon and the colour of its text to the values of self._tab_icon and self._tab_text_colour respectively
update_from_settings(settings)	

Attributes

tab_name

close_tab(***kwargs*)

get_all_save_data()

get_builtin_save_data()

get_save_data()

get_tab_layout()

initialise_GUI()

restore_save_data(*data*)

set_tab_icon_and_colour()

Set the tab icon and the colour of its text to the values of self._tab_icon and self._tab_text_colour respectively

property tab_name

update_from_settings(*settings*)

3.9.7 blacs.tab_base_classes.StateQueue

class blacs.tab_base_classes.**StateQueue**(*device_name*)

Bases: object

__init__(*device_name*)

Methods

__init__(device_name)	
check_for_next_item(state)	
get(state)	
log_current_states()	
put(allowed_states, ...[, priority])	Add a state to the queue.

Attributes

last_requested_state

check_for_next_item(*state*)

get(*state*)

property last_requested_state

log_current_states()

put(*allowed_states*, *queue_state_indefinitely*, *delete_stale_states*, *data*, *priority=0*)

Add a state to the queue. Lower number for priority indicates the state will be executed before any states with higher numbers for their priority

3.9.8 blacs.tab_base_classes.Tab

class blacs.tab_base_classes.**Tab**(*notebook*, *settings*, *restart=False*)

Bases: object

__init__(*notebook*, *settings*, *restart=False*)

Methods

__init__(notebook, settings[, restart])	
check_time()	
clean_ui_on_restart()	
close_tab([finalise])	Close the tab, terminate subprocesses and join the mainloop thread.
connect_restart_receiver(function)	
continue_restart(currentpage)	Called in a thread for the stages of restarting that may be blocking, so as to not block the main thread.
create_worker(name, WorkerClass[, workerargs])	Set up a worker process.
disconnect_restart_receiver(function)	
finalise_close_tab(currentpage)	
finalise_restart(currentpage)	
get_all_save_data()	
get_builtin_save_data()	Get builtin settings to be restored like whether the terminal is visible.
get_tab_layout()	
hide_error()	
mainloop()	
on_force_full_buffered_reprogram()	
queue_work(worker_process, worker_function, ...)	
restart(*args)	
restore_builtin_save_data(data)	Restore builtin settings to be restored like whether the terminal is visible.
set_tab_icon_and_colour()	Set the tab icon and the colour of its text to the values of self._tab_icon and self._tab_text_colour respectively
set_terminal_visible(visible)	
shutdown_workers(*args, **kwargs)	
statemachine_timeout_add(delay, ...)	
statemachine_timeout_remove(statefunction)	
statemachine_timeout_remove_all()	
supports_smart_programming(support)	
update_from_settings(settings)	

Attributes

ICON_BUSY
ICON_ERROR
ICON_FATAL_ERROR
ICON_OK
device_name
error_message
force_full_buffered_reprogram
mode
state

ICON_BUSY = ':/qtutils/fugue/hourglass'

ICON_ERROR = ':/qtutils/fugue/exclamation'

ICON_FATAL_ERROR = ':/qtutils/fugue/exclamation-red'

ICON_OK = ':/qtutils/fugue/tick'

check_time()

clean_ui_on_restart()

close_tab(*finalise=True*)

Close the tab, terminate subprocesses and join the mainloop thread. If finalise=False, then do not terminate subprocesses or join the mainloop. In this case, callers must manually call finalise_close_tab() to perform these potentially blocking operations

connect_restart_receiver(*function*)

continue_restart(*currentpage*)

Called in a thread for the stages of restarting that may be blocking, so as to not block the main thread. Calls subsequent GUI operations in the main thread once finished blocking.

create_worker(*name*, *WorkerClass*, *workerargs=None*)

Set up a worker process. WorkerClass can either be a subclass of Worker, or a string containing a fully qualified import path to a worker. The latter is useful if the worker class is in a separate file with global imports or other import-time behaviour that is undesirable to have run in the main process, for example if the imports may not be available to the main process (as may be the case once remote worker processes are implemented and the worker may be on a separate computer). The worker process will not be started immediately, it will be started once the state machine mainloop begins running. This way errors in startup will be handled using the normal state machine machinery.

property device_name

disconnect_restart_receiver(*function*)

property error_message

finalise_close_tab(*currentpage*)

finalise_restart(*currentpage*)

property force_full_buffered_reprogram

get_all_save_data()

get_builtin_save_data()

Get builtin settings to be restored like whether the terminal is visible. Not to be overridden.

get_tab_layout()

hide_error()

mainloop()

property mode

on_force_full_buffered_reprogram()

queue_work(*worker_process*, *worker_function*, **args*, ***kwargs*)

restart(**args*)

restore_builtin_save_data(*data*)

Restore builtin settings to be restored like whether the terminal is visible. Not to be overridden.

set_tab_icon_and_colour()

Set the tab icon and the colour of its text to the values of self._tab_icon and self._tab_text_colour respectively

set_terminal_visible(*visible*)

shutdown_workers(**args*, ***kwargs*)

property state

statemachine_timeout_add(*delay*, *statefunction*, **args*, ***kwargs*)

statemachine_timeout_remove(*statefunction*)

statemachine_timeout_remove_all()

supports_smart_programming(*support*)

update_from_settings(*settings*)

3.9.9 blacs.tab_base_classes.Worker

class blacs.tab_base_classes.**Worker**(**args*, ***kwargs*)

Bases: Process

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)	
init()	
interrupt_startup([reason])	Called from the parent process.
mainloop()	
run(worker_name, device_name, extraargs)	The method that gets called in the subprocess.
start(*args, **kwargs)	Call in the parent process to start a subprocess.
terminate([wait_timeout])	Interrupt process startup if not already done, ensuring self.child exists or is None if startup was interrupted before the process was created.

init()

mainloop()

run(*worker_name*, *device_name*, *extraargs*)

The method that gets called in the subprocess. To be overridden by subclasses

3.10 blacs.__main__

BLACS GUI and supporting code

Classes

BLACS(application)	
BLACSWindow	
EasterEggButton()	
ExperimentServer(*args, **kwargs)	

3.10.1 blacs.__main__.BLACS

class blacs.__main__.**BLACS**(*application*)

Bases: object

__init__(*application*)

Methods

__init__(application)
finalise_quit([deadline, pending_threads])
on_load_front_panel(*args, **kwargs)
on_open_preferences(*args, **kwargs)
on_save_exit()
on_save_front_panel(*args, **kwargs)
order_tabs(tab_data)
restore_window(tab_data)
set_relaunch(value)
update_all_tab_settings(settings, tab_data)

Attributes

tab_widget_ids

finalise_quit(*deadline=None*, *pending_threads=None*)

on_load_front_panel(**args*, ***kwargs*)

on_open_preferences(**args*, ***kwargs*)

on_save_exit()

on_save_front_panel(**args*, ***kwargs*)

order_tabs(*tab_data*)

restore_window(*tab_data*)

set_relaunch(*value*)

tab_widget_ids = 7

update_all_tab_settings(*settings*, *tab_data*)

3.10.2 blacs.__main__.BLACSWindow

class blacs.__main__.**BLACSWindow**

Bases: QMainWindow

__init__(**args*, ***kwargs*)

Methods

__init__(*args, **kwargs)
acceptDrops(self)
accessibleDescription(self)
accessibleName(self)
actionEvent(self, a0)
actions(self)
activateWindow(self)
addAction(self, action)
addActions(self, actions)
addDockWidget()
addToolBar(-> None)
addToolBarBreak(self[, area])
adjustSize(self)
autoFillBackground(self)
backgroundRole(self)
baseSize(self)
blockSignals(self, b)
centralWidget(self)

continues on next page

Table 3.6 – continued from previous page

changeEvent(self, a0)
childAt(-> Optional[QWidget])
childEvent(self, a0)
children(self)
childrenRect(self)
childrenRegion(self)
clearFocus(self)
clearMask(self)
close(self)
closeEvent(self, a0)
colorCount(self)
connectNotify(self, signal)
contentsMargins(self)
contentsRect(self)
contextMenuEvent(self, event)
contextMenuPolicy(self)
corner(self, corner)
create(self[, window, initializeWindow, ...])
createPopupMenu(self)
createWindowContainer(window[, parent, flags])
cursor(self)
customEvent(self, a0)
deleteLater(self)
depth(self)
destroy(self[, destroyWindow, destroySubWindows])
devType(self)

continues on next page

Table 3.6 – continued from previous page

devicePixelRatio(self)
devicePixelRatioF(self)
devicePixelRatioFScale()
disconnect(-> bool)
disconnectNotify(self, signal)
dockOptions(self)
dockWidgetArea(self, dockwidget)
documentMode(self)
dragEnterEvent(self, a0)
dragLeaveEvent(self, a0)
dragMoveEvent(self, a0)
dropEvent(self, a0)
dumpObjectInfo(self)
dumpObjectTree(self)
dynamicPropertyNames(self)
effectiveWinId(self)
ensurePolished(self)
enterEvent(self, a0)
event(self, event)
eventFilter(self, a0, a1)
find(a0)
findChild(-> QObjectT)
findChildren(...)
focusInEvent(self, a0)
focusNextChild(self)
focusNextPrevChild(self, next)

continues on next page

Table 3.6 – continued from previous page

focusOutEvent(self, a0)
focusPolicy(self)
focusPreviousChild(self)
focusProxy(self)
focusWidget(self)
font(self)
fontInfo(self)
fontMetrics(self)
foregroundRole(self)
frameGeometry(self)
frameSize(self)
geometry(self)
getContentsMargins(self)
grab(self[, rectangle])
grabGesture(self, type[, flags])
grabKeyboard(self)
grabMouse()
grabShortcut(self, key[, context])
graphicsEffect(self)
graphicsProxyWidget(self)
hasFocus(self)
hasHeightForWidth(self)
hasMouseTracking(self)
hasTabletTracking(self)
height(self)
heightForWidth(self, a0)

continues on next page

Table 3.6 – continued from previous page

heightMM(self)
hide(self)
hideEvent(self, a0)
iconSize(self)
inherits(self, classname)
initPainter(self, painter)
inputMethodEvent(self, a0)
inputMethodHints(self)
inputMethodQuery(self, a0)
insertAction(self, before, action)
insertActions(self, before, actions)
insertToolBar(self, before, toolbar)
insertToolBarBreak(self, before)
installEventFilter(self, a0)
isActiveWindow(self)
isAncestorOf(self, child)
isAnimated(self)
isDockNestingEnabled(self)
isEnabled(self)
isEnabledTo(self, a0)
isFullScreen(self)
isHidden(self)
isLeftToRight(self)
isMaximized(self)
isMinimized(self)
isModal(self)

continues on next page

Table 3.6 – continued from previous page

isRightToLeft(self)
isSeparator(self, pos)
isSignalConnected(self, signal)
isVisible(self)
isVisibleTo(self, a0)
isWidgetType(self)
isWindow(self)
isWindowModified(self)
isWindowType(self)
keyPressEvent(self, a0)
keyReleaseEvent(self, a0)
keyboardGrabber()
killTimer(self, id)
layout(self)
layoutDirection(self)
leaveEvent(self, a0)
locale(self)
logicalDpiX(self)
logicalDpiY(self)
lower(self)
mapFrom(self, a0, a1)
mapFromGlobal(self, a0)
mapFromParent(self, a0)
mapTo(self, a0, a1)
mapToGlobal(self, a0)
mapToParent(self, a0)

continues on next page

Table 3.6 – continued from previous page

mask(self)
maximumHeight(self)
maximumSize(self)
maximumWidth(self)
menuBar(self)
menuWidget(self)
metaObject(self)
metric(self, a0)
minimumHeight(self)
minimumSize(self)
minimumSizeHint(self)
minimumWidth(self)
mouseDoubleClickEvent(self, a0)
mouseGrabber()
mouseMoveEvent(self, a0)
mousePressEvent(self, a0)
mouseReleaseEvent(self, a0)
move()
moveEvent(self, a0)
moveToThread(self, thread)
nativeEvent(self, eventType, message)
nativeParentWidget(self)
nextInFocusChain(self)
normalGeometry(self)
objectName(self)
overrideWindowFlags(self, type)

continues on next page

Table 3.6 – continued from previous page

overrideWindowState(self, state)	
paintEngine(self)	
paintEvent(self, a0)	
paintingActive(self)	
palette(self)	
parent(self)	
parentWidget(self)	
physicalDpiX(self)	
physicalDpiY(self)	
pos(self)	
previousInFocusChain(self)	
property(self, name)	
pyqtConfigure(...)	Each keyword argument is either the name of a Qt property or a Qt signal.
raise_(self)	
receivers(self, signal)	
rect(self)	
releaseKeyboard(self)	
releaseMouse(self)	
releaseShortcut(self, id)	
removeAction(self, action)	
removeDockWidget(self, dockwidget)	
removeEventFilter(self, a0)	
removeToolBar(self, toolbar)	
removeToolBarBreak(self, before)	
render(, sourceRegion, flags, ...)	
repaint(-> None -> None)	

continues on next page

Table 3.6 – continued from previous page

resize()
resizeDocks(self, docks, sizes, orientation)
resizeEvent(self, a0)
restoreDockWidget(self, dockwidget)
restoreGeometry(self, geometry)
restoreState(self, state[, version])
saveGeometry(self)
saveState(self[, version])
screen(self)
scroll()
sender(self)
senderSignalIndex(self)
setAcceptDrops(self, on)
setAccessibleDescription(self, description)
setAccessibleName(self, name)
setAnimated(self, enabled)
setAttribute(self, attribute[, on])
setAutoFillBackground(self, enabled)
setBackgroundRole(self, a0)
setBaseSize()
setCentralWidget(self, widget)
setContentsMargins()
setContextMenuPolicy(self, policy)
setCorner(self, corner, area)
setCursor(self, a0)
setDisabled(self, a0)

continues on next page

Table 3.6 – continued from previous page

setDockNestingEnabled(self, enabled)
setDockOptions(self, options)
setDocumentMode(self, enabled)
setEnabled(self, a0)
setFixedHeight(self, h)
setFixedSize()
setFixedWidth(self, w)
setFocus()
setFocusPolicy(self, policy)
setFocusProxy(self, a0)
setFont(self, a0)
setForegroundRole(self, a0)
setGeometry()
setGraphicsEffect(self, effect)
setHidden(self, hidden)
setIconSize(self, iconSize)
setInputMethodHints(self, hints)
setLayout(self, a0)
setLayoutDirection(self, direction)
setLocale(self, locale)
setMask()
setMaximumHeight(self, maxh)
setMaximumSize()
setMaximumWidth(self, maxw)
setMenuBar(self, menubar)
setMenuWidget(self, menubar)

continues on next page

Table 3.6 – continued from previous page

setMinimumHeight(self, minh)
setMinimumSize()
setMinimumWidth(self, minw)
setMouseTracking(self, enable)
setObjectName(self, name)
setPalette(self, a0)
setParent()
setProperty(self, name, value)
setShortcutAutoRepeat(self, id[, enabled])
setShortcutEnabled(self, id[, enabled])
setSizeIncrement()
setSizePolicy()
setStatusBar(self, statusbar)
setStatusTip(self, a0)
setStyle(self, a0)
setStyleSheet(self, styleSheet)
setTabOrder(a0, a1)
setTabPosition(self, areas, tabPosition)
setTabShape(self, tabShape)
setTabletTracking(self, enable)
setToolButtonStyle(self, toolButtonStyle)
setToolTip(self, a0)
setToolTipDuration(self, msec)
setUnifiedTitleAndToolBarOnMac(self, set)
setUpdatesEnabled(self, enable)
setVisible(self, visible)

continues on next page

Table 3.6 – continued from previous page

setWhatsThis(self, a0)
setWindowFilePath(self, filePath)
setWindowFlag(self, a0[, on])
setWindowFlags(self, type)
setWindowIcon(self, icon)
setWindowIconText(self, a0)
setWindowModality(self, windowModality)
setWindowModified(self, a0)
setWindowOpacity(self, level)
setWindowRole(self, a0)
setWindowState(self, state)
setWindowTitle(self, a0)
sharedPainter(self)
show(self)
showEvent(self, a0)
showFullScreen(self)
showMaximized(self)
showMinimized(self)
showNormal(self)
signalsBlocked(self)
size(self)
sizeHint(self)
sizeIncrement(self)
sizePolicy(self)
splitDockWidget(self, after, dockwidget, ...)
stackUnder(self, a0)

continues on next page

Table 3.6 – continued from previous page

startTimer(self, interval[, timerType])
statusBar(self)
statusTip(self)
style(self)
styleSheet(self)
tabPosition(self, area)
tabShape(self)
tabifiedDockWidgets(self, dockwidget)
tabifyDockWidget(self, first, second)
tabletEvent(self, a0)
takeCentralWidget(self)
testAttribute(self, attribute)
thread(self)
timerEvent(self, a0)
toolBarArea(self, toolbar)
toolBarBreak(self, toolbar)
toolButtonStyle(self)
toolTip(self)
toolTipDuration(self)
tr(self, sourceText[, disambiguation, n])
underMouse(self)
ungrabGesture(self, type)
unifiedTitleAndToolBarOnMac(self)
unsetCursor(self)
unsetLayoutDirection(self)
unsetLocale(self)

continues on next page

Table 3.6 – continued from previous page

update(-> None -> None)
updateGeometry(self)
updateMicroFocus(self)
updatesEnabled(self)
visibleRegion(self)
whatsThis(self)
wheelEvent(self, a0)
width(self)
widthMM(self)
winId(self)
window(self)
windowFilePath(self)
windowFlags(self)
windowHandle(self)
windowIcon(self)
windowIconText(self)
windowModality(self)
windowOpacity(self)
windowRole(self)
windowState(self)
windowTitle(self)
windowType(self)
x(self)
y(self)

Attributes

AllowNestedDocks	
AllowTabbedDocks	
AnimatedDocks	
DrawChildren	
DrawWindowBackground	
ForceTabbedDocks	
GroupedDragging	
IgnoreMask	
PdmDepth	
PdmDevicePixelRatio	
PdmDevicePixelRatioScaled	
PdmDpiX	
PdmDpiY	
PdmHeight	
PdmHeightMM	
PdmNumColors	
PdmPhysicalDpiX	
PdmPhysicalDpiY	
PdmWidth	
PdmWidthMM	
VerticalTabs	
customContextMenuRequested	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
destroyed	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
iconSizeChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
objectNameChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL

continues on next page

Table 3.7 – continued from previous page

staticMetaObject	
tabifiedDockWidgetActivated	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
toolButtonStyleChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
windowIconChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
windowIconTextChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
windowTitleChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL

closeEvent(*self*, *a0: QCloseEvent | None*)

3.10.3 blacs.__main__.EasterEggButton

class blacs.__main__.**EasterEggButton**

Bases: QToolButton

__init__()

Methods

__init__()
acceptDrops(self)
accessibleDescription(self)
accessibleName(self)
actionEvent(self, a0)
actions(self)
activateWindow(self)
addAction(self, action)
addActions(self, actions)
adjustSize(self)
animateClick(self[, msecs])
arrowType(self)

continues on next page

Table 3.8 – continued from previous page

autoExclusive(self)
autoFillBackground(self)
autoRaise(self)
autoRepeat(self)
autoRepeatDelay(self)
autoRepeatInterval(self)
backgroundRole(self)
baseSize(self)
blockSignals(self, b)
changeEvent(self, a0)
checkStateSet(self)
childAt(-> Optional[QWidget])
childEvent(self, a0)
children(self)
childrenRect(self)
childrenRegion(self)
clearFocus(self)
clearMask(self)
click(self)
close(self)
closeEvent(self, a0)
colorCount(self)
connectNotify(self, signal)
contentsMargins(self)
contentsRect(self)
contextMenuEvent(self, a0)

continues on next page

Table 3.8 – continued from previous page

contextMenuPolicy(self)	
create(self[, window, initializeWindow, ...])	
createWindowContainer(window[, parent, flags])	
cursor(self)	
customEvent(self, a0)	
defaultAction(self)	
deleteLater(self)	
depth(self)	
destroy(self[, destroyWindow, destroySubWindows])	
devType(self)	
devicePixelRatio(self)	
devicePixelRatioF(self)	
devicePixelRatioFScale()	
disconnect(-> bool)	
disconnectNotify(self, signal)	
dragEnterEvent(self, a0)	
dragLeaveEvent(self, a0)	
dragMoveEvent(self, a0)	
dropEvent(self, a0)	
dumpObjectInfo(self)	
dumpObjectTree(self)	
dynamicPropertyNames(self)	
effectiveWinId(self)	
ensurePolished(self)	
enterEvent(event)	Make the icon only visible on mouse-over
event(self, e)	

continues on next page

Table 3.8 – continued from previous page

eventFilter(self, a0, a1)
find(a0)
findChild(-> QObjectT)
findChildren(...)
focusInEvent(self, e)
focusNextChild(self)
focusNextPrevChild(self, next)
focusOutEvent(self, e)
focusPolicy(self)
focusPreviousChild(self)
focusProxy(self)
focusWidget(self)
font(self)
fontInfo(self)
fontMetrics(self)
foregroundRole(self)
frameGeometry(self)
frameSize(self)
geometry(self)
getContentsMargins(self)
grab(self[, rectangle])
grabGesture(self, type[, flags])
grabKeyboard(self)
grabMouse()
grabShortcut(self, key[, context])
graphicsEffect(self)

continues on next page

Table 3.8 – continued from previous page

graphicsProxyWidget(self)
group(self)
hasFocus(self)
hasHeightForWidth(self)
hasMouseTracking(self)
hasTabletTracking(self)
height(self)
heightForWidth(self, a0)
heightMM(self)
hide(self)
hideEvent(self, a0)
hitButton(self, pos)
icon(self)
iconSize(self)
inherits(self, classname)
initPainter(self, painter)
initStyleOption(self, option)
inputMethodEvent(self, a0)
inputMethodHints(self)
inputMethodQuery(self, a0)
insertAction(self, before, action)
insertActions(self, before, actions)
installEventFilter(self, a0)
isActiveWindow(self)
isAncestorOf(self, child)
isCheckable(self)

continues on next page

Table 3.8 – continued from previous page

isChecked(self)
isDown(self)
isEnabled(self)
isEnabledTo(self, a0)
isFullScreen(self)
isHidden(self)
isLeftToRight(self)
isMaximized(self)
isMinimized(self)
isModal(self)
isRightToLeft(self)
isSignalConnected(self, signal)
isVisible(self)
isVisibleTo(self, a0)
isWidgetType(self)
isWindow(self)
isWindowModified(self)
isWindowType(self)
keyPressEvent(self, e)
keyReleaseEvent(self, e)
keyboardGrabber()
killTimer(self, id)
layout(self)
layoutDirection(self)
leaveEvent(self, a0)
locale(self)

continues on next page

Table 3.8 – continued from previous page

logicalDpiX(self)
logicalDpiY(self)
lower(self)
mapFrom(self, a0, a1)
mapFromGlobal(self, a0)
mapFromParent(self, a0)
mapTo(self, a0, a1)
mapToGlobal(self, a0)
mapToParent(self, a0)
mask(self)
maximumHeight(self)
maximumSize(self)
maximumWidth(self)
menu(self)
metaObject(self)
metric(self, a0)
minimumHeight(self)
minimumSize(self)
minimumSizeHint(self)
minimumWidth(self)
mouseDoubleClickEvent(self, a0)
mouseGrabber()
mouseMoveEvent(self, e)
mousePressEvent(self, a0)
mouseReleaseEvent(self, a0)
move()

continues on next page

Table 3.8 – continued from previous page

moveEvent(self, a0)	
moveToThread(self, thread)	
nativeEvent(self, eventType, message)	
nativeParentWidget(self)	
nextCheckState(self)	
nextInFocusChain(self)	
normalGeometry(self)	
objectName(self)	
on_click()	Run Measure Ball
overrideWindowFlags(self, type)	
overrideWindowState(self, state)	
paintEngine(self)	
paintEvent(self, a0)	
paintingActive(self)	
palette(self)	
parent(self)	
parentWidget(self)	
physicalDpiX(self)	
physicalDpiY(self)	
popupMode(self)	
pos(self)	
previousInFocusChain(self)	
property(self, name)	
pyqtConfigure(...)	Each keyword argument is either the name of a Qt property or a Qt signal.
raise_(self)	
receivers(self, signal)	

continues on next page

Table 3.8 – continued from previous page

rect(self)
releaseKeyboard(self)
releaseMouse(self)
releaseShortcut(self, id)
removeAction(self, action)
removeEventFilter(self, a0)
render(, sourceRegion, flags, ...)
repaint(-> None -> None)
resize()
resizeEvent(self, a0)
restoreGeometry(self, geometry)
run_measure_ball()
saveGeometry(self)
screen(self)
scroll()
sender(self)
senderSignalIndex(self)
setAcceptDrops(self, on)
setAccessibleDescription(self, description)
setAccessibleName(self, name)
setArrowType(self, type)
setAttribute(self, attribute[, on])
setAutoExclusive(self, a0)
setAutoFillBackground(self, enabled)
setAutoRaise(self, enable)
setAutoRepeat(self, a0)

continues on next page

Table 3.8 – continued from previous page

setAutoRepeatDelay(self, a0)
setAutoRepeatInterval(self, a0)
setBackgroundRole(self, a0)
setBaseSize()
setCheckable(self, a0)
setChecked(self, a0)
setContentsMargins()
setContextMenuPolicy(self, policy)
setCursor(self, a0)
setDefaultAction(self, a0)
setDisabled(self, a0)
setDown(self, a0)
setEnabled(self, a0)
setFixedHeight(self, h)
setFixedSize()
setFixedWidth(self, w)
setFocus()
setFocusPolicy(self, policy)
setFocusProxy(self, a0)
setFont(self, a0)
setForegroundRole(self, a0)
setGeometry()
setGraphicsEffect(self, effect)
setHidden(self, hidden)
setIcon(self, icon)
setIconSize(self, size)

continues on next page

Table 3.8 – continued from previous page

setInputMethodHints(self, hints)
setLayout(self, a0)
setLayoutDirection(self, direction)
setLocale(self, locale)
setMask()
setMaximumHeight(self, maxh)
setMaximumSize()
setMaximumWidth(self, maxw)
setMenu(self, menu)
setMinimumHeight(self, minh)
setMinimumSize()
setMinimumWidth(self, minw)
setMouseTracking(self, enable)
setObjectName(self, name)
setPalette(self, a0)
setParent()
setPopupMode(self, mode)
setProperty(self, name, value)
setShortcut(self, key)
setShortcutAutoRepeat(self, id[, enabled])
setShortcutEnabled(self, id[, enabled])
setSizeIncrement()
setSizePolicy()
setStatusTip(self, a0)
setStyle(self, a0)
setStyleSheet(self, styleSheet)

continues on next page

Table 3.8 – continued from previous page

setTabOrder(a0, a1)
setTabletTracking(self, enable)
setText(self, text)
setToolButtonStyle(self, style)
setToolTip(self, a0)
setToolTipDuration(self, msec)
setUpdatesEnabled(self, enable)
setVisible(self, visible)
setWhatsThis(self, a0)
setWindowFilePath(self, filePath)
setWindowFlag(self, a0[, on])
setWindowFlags(self, type)
setWindowIcon(self, icon)
setWindowIconText(self, a0)
setWindowModality(self, windowModality)
setWindowModified(self, a0)
setWindowOpacity(self, level)
setWindowRole(self, a0)
setWindowState(self, state)
setWindowTitle(self, a0)
sharedPainter(self)
shortcut(self)
show(self)
showEvent(self, a0)
showFullScreen(self)
showMaximized(self)

continues on next page

Table 3.8 – continued from previous page

showMenu(self)
showMinimized(self)
showNormal(self)
signalsBlocked(self)
size(self)
sizeHint(self)
sizeIncrement(self)
sizePolicy(self)
stackUnder(self, a0)
startTimer(self, interval[, timerType])
statusTip(self)
style(self)
styleSheet(self)
tabletEvent(self, a0)
testAttribute(self, attribute)
text(self)
thread(self)
timerEvent(self, a0)
toggle(self)
toolButtonStyle(self)
toolTip(self)
toolTipDuration(self)
tr(self, sourceText[, disambiguation, n])
underMouse(self)
ungrabGesture(self, type)
unsetCursor(self)

continues on next page

Table 3.8 – continued from previous page

unsetLayoutDirection(self)
unsetLocale(self)
update(-> None -> None)
updateGeometry(self)
updateMicroFocus(self)
updatesEnabled(self)
visibleRegion(self)
whatsThis(self)
wheelEvent(self, a0)
width(self)
widthMM(self)
winId(self)
window(self)
windowFilePath(self)
windowFlags(self)
windowHandle(self)
windowIcon(self)
windowIconText(self)
windowModality(self)
windowOpacity(self)
windowRole(self)
windowState(self)
windowTitle(self)
windowType(self)
x(self)

continues on next page

Table 3.8 – continued from previous page

y(self)	

Attributes

DelayedPopup	
DrawChildren	
DrawWindowBackground	
IgnoreMask	
InstantPopup	
MenuButtonPopup	
PdmDepth	
PdmDevicePixelRatio	
PdmDevicePixelRatioScaled	
PdmDpiX	
PdmDpiY	
PdmHeight	
PdmHeightMM	
PdmNumColors	
PdmPhysicalDpiX	
PdmPhysicalDpiY	
PdmWidth	
PdmWidthMM	
clicked	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
customContextMenuRequested	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
destroyed	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
objectNameChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
pressed	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
released	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
staticMetaObject	
toggled	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
triggered	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
windowIconChanged	int = ..., arguments: Sequence = ...) -> PYQT_SIGNAL
windowIconTextChanged	int = ..., arguments: Sequence = ...) ->

enterEvent(*event*)

Make the icon only visible on mouse-over

leaveEvent(*self*, *a0: QEvent | None*)

on_click()

Run Measure Ball

run_measure_ball()

3.10.4 blacs.__main__.ExperimentServer

class blacs.__main__.**ExperimentServer**(**args*, ***kwargs*)

Bases: ZMQServer

Parameters

- **args** (*Any*) –
- **kwargs** (*Any*) –

Return type
Any

__init__(**args*, ***kwargs*)

Parameters

- **args** (*Any*) –
- **kwargs** (*Any*) –

Return type
None

Methods

__init__(*args, **kwargs)
handler(h5_filepath)
process(h5_filepath)

handler(*h5_filepath*)

process(*h5_filepath*)

CHAPTER

FOUR

LABSCRIPT SUITE COMPONENTS

The *labscript suite* is modular by design, and is comprised of:

Table 4.1: Python libraries

labscript — Expressive composition of hardware-timed experiments
labscript-devices — Plugin architecture for controlling experiment hardware
labscript-utils — Shared modules used by the *labscript suite*

Table 4.2: Graphical applications

runmanager — Graphical and remote interface to parameterized experiments
blacs — Graphical interface to scientific instruments and experiment supervision
lyse — Online analysis of live experiment data
runviewer — Visualize hardware-timed experiment instructions

PYTHON MODULE INDEX

b

INDEX

Symbols

A

B

C

D

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W