

DvG_QDeviceIO.py

Dependencies:

enum
queue
time
numpy
PyQt5
DvG_debug_functions

<<PyQt5.QtCore.QObject>>

QDeviceIO

```
<<PyQt5.QtCore.pyqtSignal>>
signal_DAQ_updated()
signal_DAQ_suspended()
signal_connection_lost()
-----
dev          : {linked I/O device class}
dev.name     : str
dev.mutex    : PyQt5.QtCore.QMutex()
dev.is_alive : bool
_thread_DAQ  : PyQt5.QtCore.QThread()
_thread_send : PyQt5.QtCore.QThread()
worker_DAQ   : Worker_DAQ()
worker_send  : Worker_send()
DAQ_update_counter
DAQ_not_alive_counter
obtained_DAQ_update_interval_ms
obtained_DAQ_rate_Hz
-----
__init__()
attach_device(dev)
create_worker_DAQ(**kwargs)
create_worker_send(**kwargs)
start_worker_DAQ(
    priority : PyQt5.QtCore.QThread.Priority)
start_worker_send(
    priority : PyQt5.QtCore.QThread.Priority)
quit_worker_DAQ()
quit_worker_send()
quit_all_workers()
```

<<object>>

InnerClassDescriptor

```
cls
outer
-----
__init__(cls)
__get__(instance, outerclass)
```

@InnerClassDescriptor

<<PyQt5.QtCore.QObject>>

QDeviceIO.Worker_send

```
DEBUG      : bool
DEBUG_color
dev         : {linked I/O device class}
alt_process_jobs_function : function
update_counter : int
qwc         : PyQt5.QtCore.QWaitCondition()
mutex_wait  : PyQt5.QtCore.QMutex()
running     : bool
sentinel
queue       : queue.Queue()
-----
__init__(
    alt_process_jobs_function : function,
    DEBUG : bool)
_do_work()
_stop()
add_to_queue(instruction, pass_args)
process_queue()
queued_instruction(instruction, pass_args)
```

@InnerClassDescriptor

<<PyQt5.QtCore.QObject>>

QDeviceIO.Worker_DAQ

```
DEBUG      : bool
DEBUG_color
dev         : {linked I/O device class}
update_interval_ms
function_to_run_each_update : function
critical_not_alive_count
timer_type : PyQt5.QtCore.Qt.TimerType
trigger_by : DAQ_trigger
qwc         : PyQt5.QtCore.QWaitCondition()
mutex_wait  : PyQt5.QtCore.QMutex()
running     : bool
suspend     : bool
suspended   : bool
calc_DAQ_rate_every_N_iter
QET_DAQ     : PyQt5.QtCore.QElapsedTimer()
prev_tick_DAQ_update
prev_tick_DAQ_rate
-----
__init__(
    DAQ_trigger_by : DAQ_trigger,
    DAQ_function_to_run_each_update : function,
    DAQ_update_interval_ms,
    DAQ_timer_type : PyQt5.QtCore.Qt.TimerType,
    DAQ_critical_not_alive_count,
    calc_DAQ_rate_every_N_iter,
    DEBUG : bool)
_do_work()
_stop()
_perform_DAQ()
schedule_suspend(state : bool)
wake_up()
```

@enum.unique

<<enum.IntEnum>>

DAQ_trigger

INTERNAL_TIMER
SINGLE_SHOT_WAKE_UP
CONTINUOUS