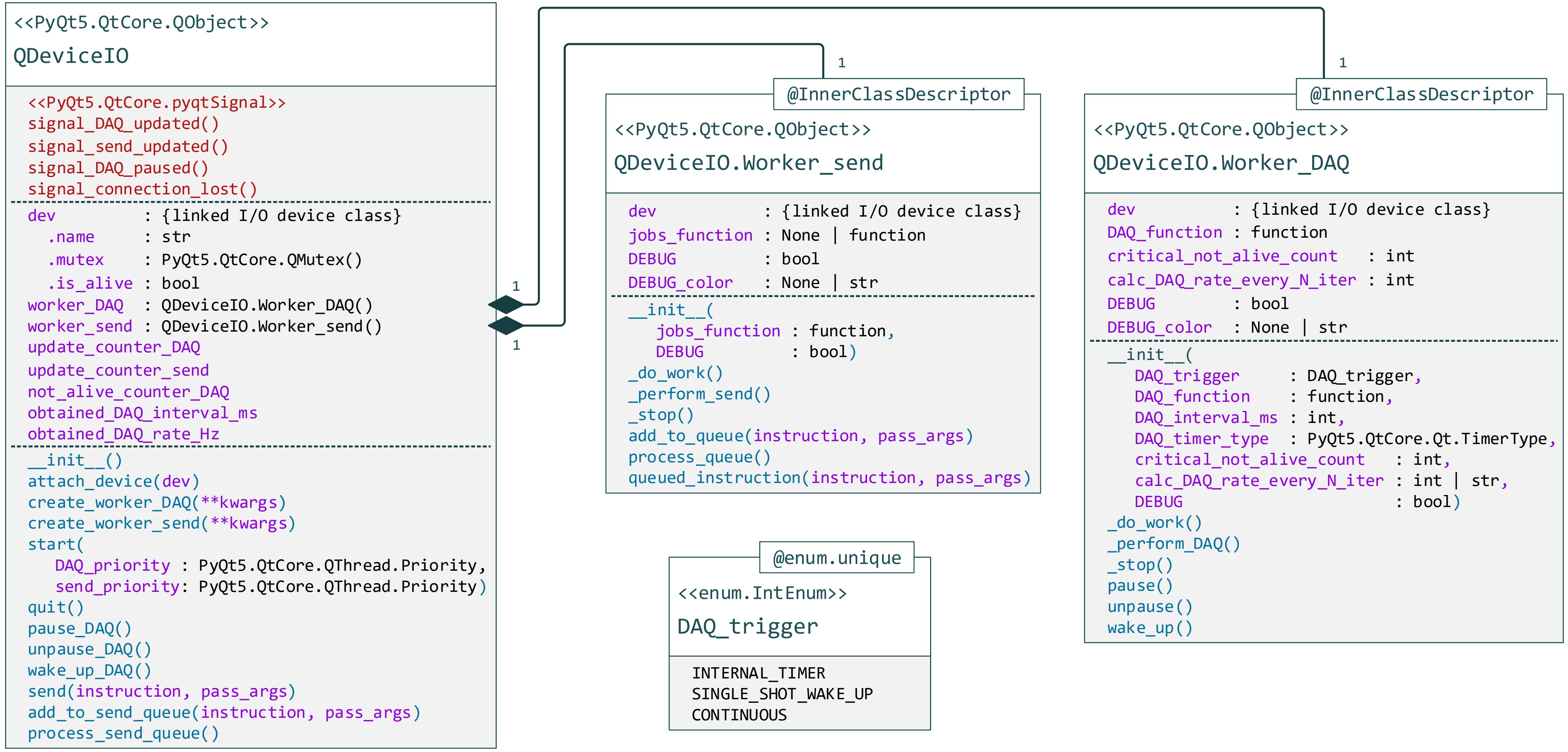
DvG_QDeviceIO.py

v0.0.8

```
Dependencies:
 numpy
 PyQt5
 DvG_debug_functions
```



```
<<PyQt5.QtCore.QObject>>
QDeviceIO
-------------------------------------------------
<<PyQt5.QtCore.pyqtSignal>>
signal_DAQ_updated()
signal_send_updated()
signal_DAQ_paused()
signal_connection_lost()
.................................................
dev          : {linked I/O device class}
  .name      : str
  .mutex     : PyQt5.QtCore.QMutex()
  .is_alive  : bool
worker_DAQ   : QDeviceIO.Worker_DAQ()
worker_send  : QDeviceIO.Worker_send()
update_counter_DAQ
update_counter_send
not_alive_counter_DAQ
obtained_DAQ_interval_ms
obtained_DAQ_rate_Hz
.................................................
__init__()
attach_device(dev)
create_worker_DAQ(**kwargs)
create_worker_send(**kwargs)
start(
   DAQ_priority : PyQt5.QtCore.QThread.Priority,
   send_priority: PyQt5.QtCore.QThread.Priority)
quit()
pause_DAQ()
unpause_DAQ()
wake_up_DAQ()
send(instruction, pass_args)
add_to_send_queue(instruction, pass_args)
process_send_queue()
```

```
@InnerClassDescriptor
<<PyQt5.QtCore.QObject>>
QDeviceIO.Worker_send
-------------------------------------------------
dev            : {linked I/O device class}
jobs_function  : None | function
DEBUG          : bool
DEBUG_color    : None | str
.................................................
__init__(
   jobs_function  : function,
   DEBUG          : bool)
_do_work()
_perform_send()
_stop()
add_to_queue(instruction, pass_args)
process_queue()
queued_instruction(instruction, pass_args)
```

```
@InnerClassDescriptor
<<PyQt5.QtCore.QObject>>
QDeviceIO.Worker_DAQ
-------------------------------------------------
dev           : {linked I/O device class}
DAQ_function  : function
critical_not_alive_count   : int
calc_DAQ_rate_every_N_iter : int
DEBUG         : bool
DEBUG_color   : None | str
.................................................
__init__(
   DAQ_trigger      : DAQ_trigger,
   DAQ_function     : function,
   DAQ_interval_ms  : int,
   DAQ_timer_type   : PyQt5.QtCore.Qt.TimerType,
   critical_not_alive_count   : int,
   calc_DAQ_rate_every_N_iter : int | str,
   DEBUG                      : bool)
_do_work()
_perform_DAQ()
_stop()
pause()
unpause()
wake_up()
```

```
@enum.unique
<<enum.IntEnum>>
DAQ_trigger
-------------------------------------------------
INTERNAL_TIMER
SINGLE_SHOT_WAKE_UP
CONTINUOUS
```