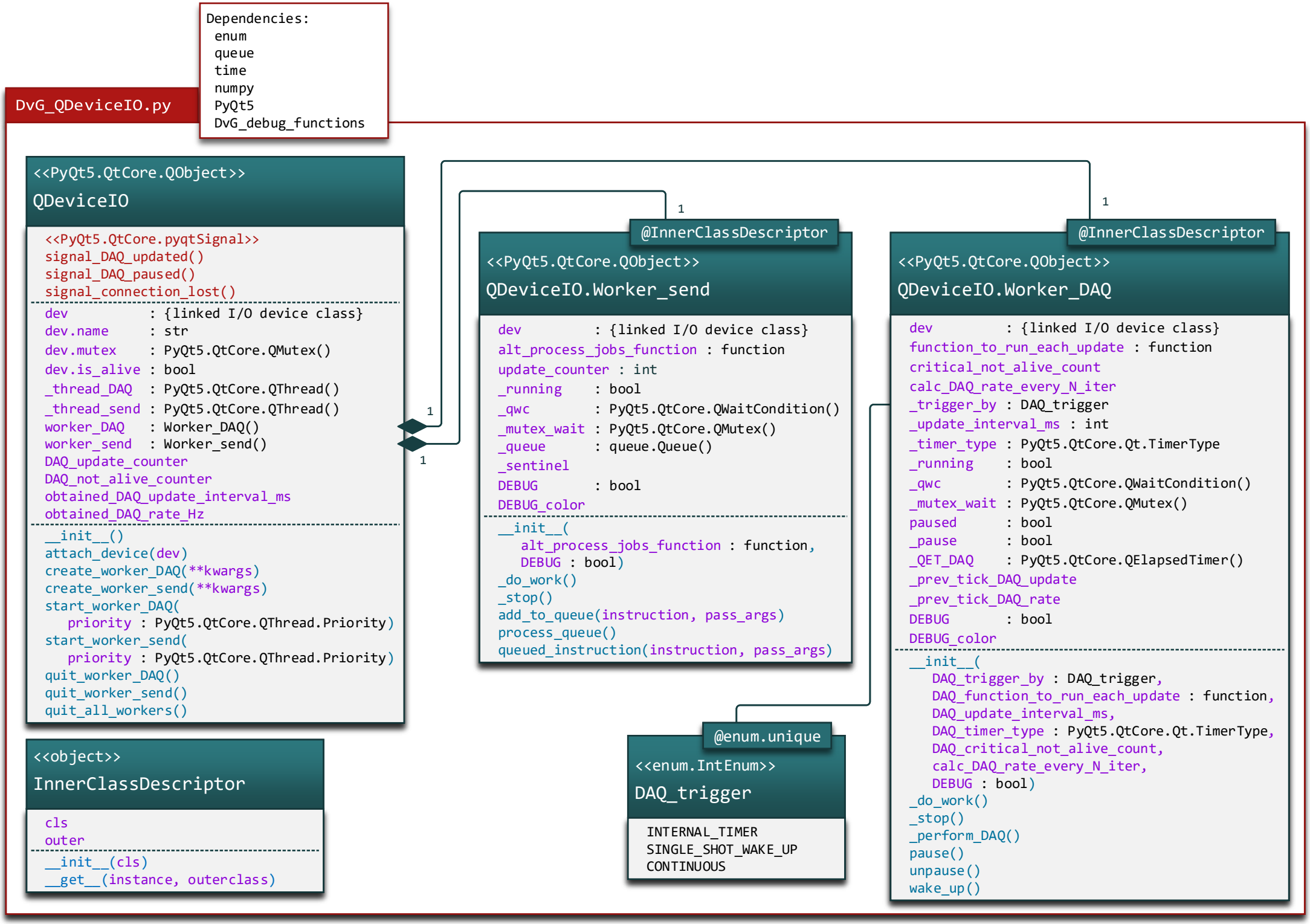
Dependencies:
enum
queue
time
numpy
PyQt5
DvG_debug_functions
DvG_QDeviceIO.py
<<PyQt5.QtCore.QObject>>
QDeviceIO
<<PyQt5.pyqtSignal>>
signal_DAQ_updated()
signal_DAQ_paused()
signal_connection_lost()
dev : {linked I/O device class}
dev.name : str
dev.mutex : PyQt5.QtCore.QMutex()
dev.is_alive : bool
_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()
1
1
1
1
@InnerClassDescriptor
<<PyQt5.QtCore.QObject>>
QDeviceIO.Worker_send
dev : {linked I/O device class}
alt_process_jobs_function : function
update_counter : int
_running : bool
_qwc : PyQt5.QtCore.QWaitCondition()
_mutex_wait : PyQt5.QtCore.QMutex()
_queue : queue.Queue()
_sentinel
DEBUG : bool
DEBUG_color
__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
dev : {linked I/O device class}
function_to_run_each_update : function
critical_not_alive_count
calc_DAQ_rate_every_N_iter
_trigger_by : DAQ_trigger
_update_interval_ms : int
_timer_type : PyQt5.QtCore.Qt.TimerType
_running : bool
_qwc : PyQt5.QtCore.QWaitCondition()
_mutex_wait : PyQt5.QtCore.QMutex()
paused : bool
_pause : bool
_QET_DAQ : PyQt5.QtCore.QElapsedTimer()
_prev_tick_DAQ_update
_prev_tick_DAQ_rate
DEBUG : bool
DEBUG_color
__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()
pause()
unpause()
wake_up()
@enum.unique
<<enum.IntEnum>>
DAQ_trigger
INTERNAL_TIMER
SINGLE_SHOT_WAKE_UP
CONTINUOUS
<<object>>
InnerClassDescriptor
cls
outer
__init__(cls)
__get__(instance, outerclass)