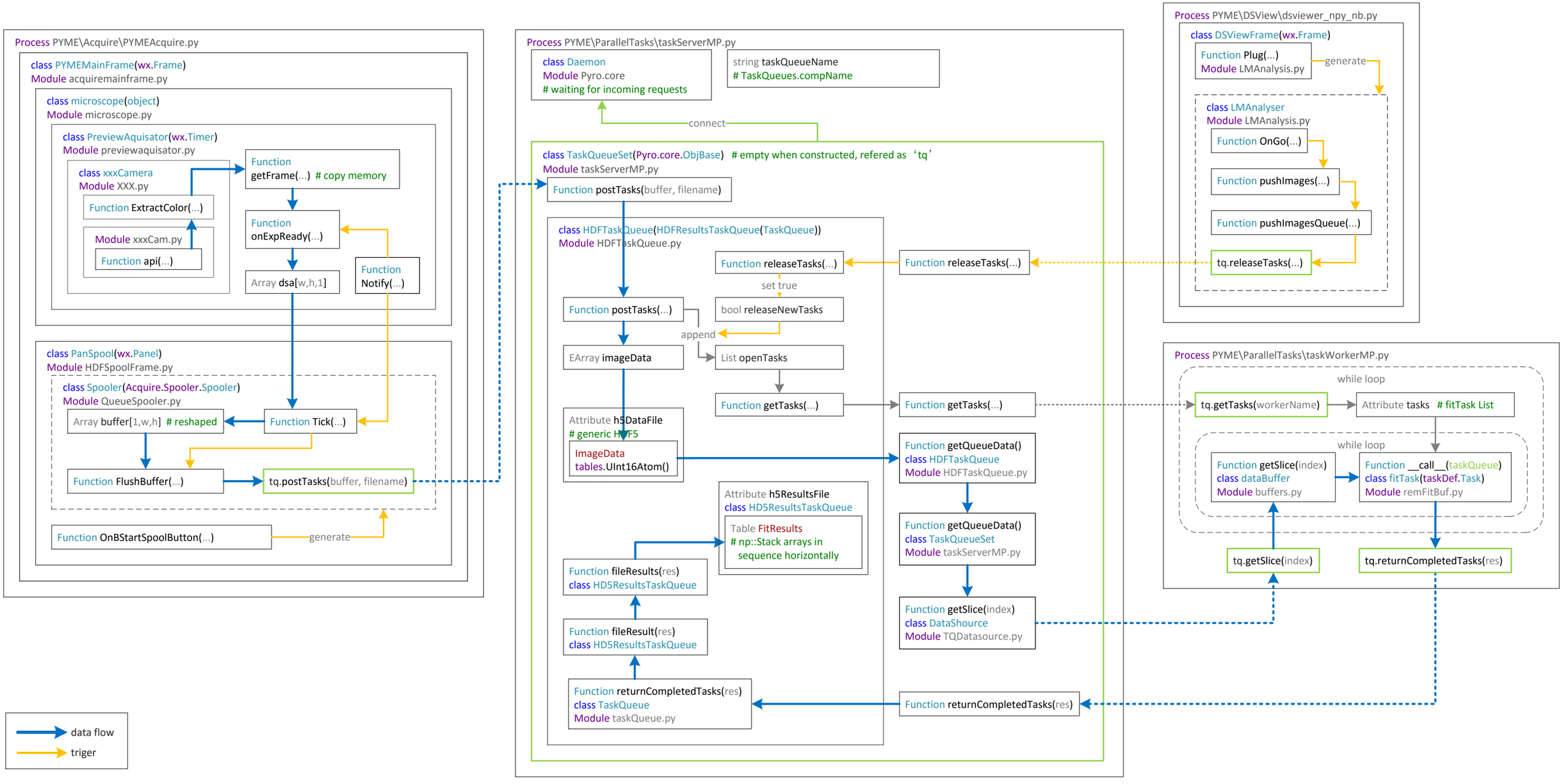
Process PYME\Acquire\PYMEAcquire.py
class PYMEMainFrame(wx.Frame)
Module acquiremainframe.py
class microscope(object)
Module microscope.py
class PreviewAquisator(wx.Timer)
Module previewaquisator.py
class xxxCamera
Module XXX.py
Function ExtractColor(...)
Module xxxCam.py
Function api(...)
Function getFrame(...) # copy memory
Function onExpReady(...)
Array dsa[w,h,1]
Function Notify(...)
class PanSpool(wx.Panel)
Module HDFSpoolFrame.py
class Spooler(Acquire.Spooler.Spooler)
Module QueueSpooler.py
Array buffer[1,w,h] # reshaped
Function Tick(...)
Function FlushBuffer(...)
tq.postTasks(buffer, filename)
Function OnBStartSpoolButton(...)
generate
Process PYME\ParallelTasks\taskServerMP.py
class Daemon
Module Pyro.core
waiting for incoming requests
string taskQueueName
TaskQueues.compName
connect
class TaskQueueSet(Pyro.core.ObjBase) # empty when constructed, refered as 'tq'
Module taskServerMP.py
Function postTasks(buffer, filename)
class HDFTaskQueue(HDFResultsTaskQueue(TaskQueue))
Module HDFTaskQueue.py
Function releaseTasks(...)
set true
bool releaseNewTasks
Function postTasks(...)
append
EArray imageData
List openTasks
Function getTasks(...)
Attribute h5DataFile
generic HDF5
ImageData
tables.UInt16Atom()
Attribute h5ResultsFile
class HD5ResultsTaskQueue
Table FitResults
np::Stack arrays in
sequence horizontally
Function fileResults(res)
class HD5ResultsTaskQueue
Function fileResult(res)
class HD5ResultsTaskQueue
Function returnCompletedTasks(res)
class TaskQueue
Module taskQueue.py
Function releaseTasks(...)
Function getTasks(...)
Function getQueueData()
class HDFTaskQueue
Module HDFTaskQueue.py
Function getQueueData()
class TaskQueueSet
Module taskServerMP.py
Function getSlice(index)
class DataShource
Module TQDatasource.py
Function returnCompletedTasks(res)
Process PYME\DSView\dsviewer_npy_nb.py
class DSViewFrame(wx.Frame)
Function Plug(...)
Module LMAnalysis.py
generate
class LMAnalyser
Module LMAnalysis.py
Function OnGo(...)
Function pushImages(...)
Function pushImagesQueue(...)
tq.releaseTasks(...)
Process PYME\ParallelTasks\taskWorkerMP.py
while loop
tq.getTasks(workerName)
Attribute tasks # fitTask List
while loop
Function getSlice(index)
class dataBuffer
Module buffers.py
Function __call__(taskQueue)
class fitTask(taskDef.Task)
Module remFitBuf.py
tq.getSlice(index)
tq.returnCompletedTasks(res)
data flow
triger