Control microscope

2017, Maxime PINSARD

To you, who wants to modify/improve the python code

- Be aware that the main GUI should NOT call an inner function for doing an action that is not instantaneous, it should rather emit a pyQtSignal that is connected to another worker that will perform the action

→ For instance, to home the XY stage (takes 3 secs) : you should tell the APT worker to do it in parallel (so the APT libs and motors should be initialised in this worker)

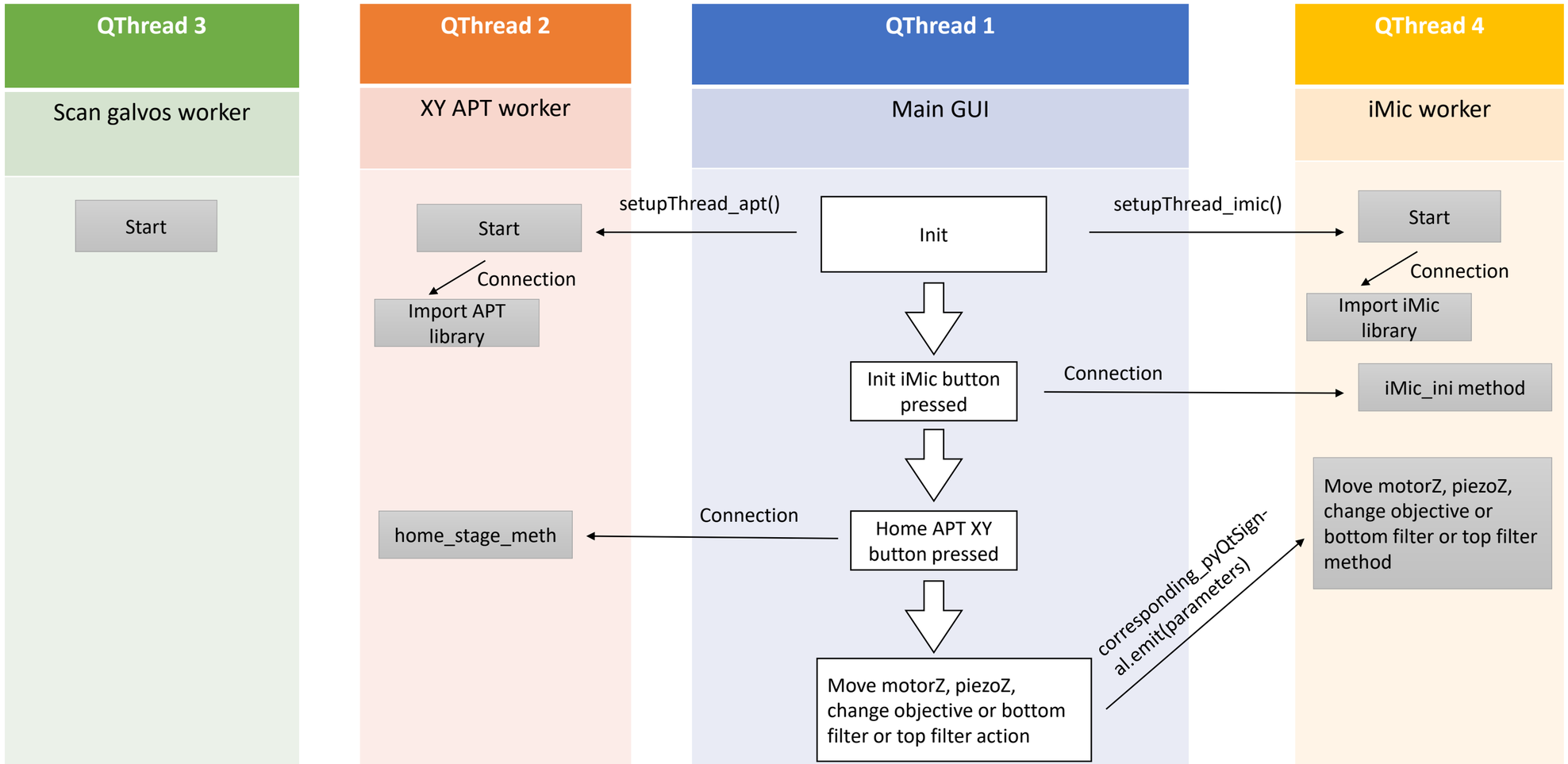
→ By clicking “home button”, you should be connected directly to one of the APT worker’s method: if you go through a GUI method before, it will wait inside this method until it’s completed and will freeze the GUI

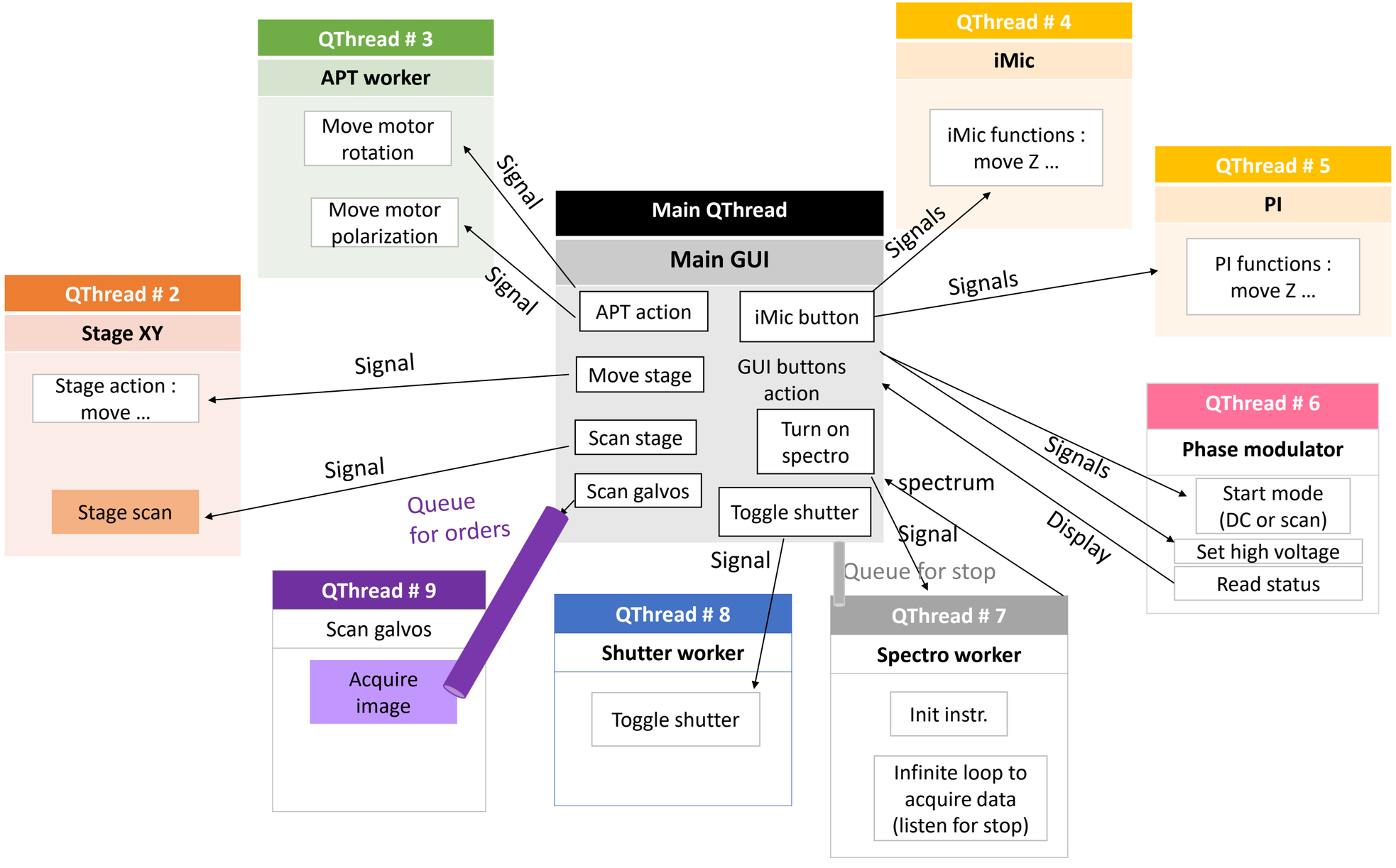
- For multiprocess, non ‘normal’ variable are said ‘non-pickable’ : you should only pass classic python and numpy objects to the init of processes. This means you can’t pass VISA resource, DAQ object nor APT motor nor any library to the init, you rather have to init them inside the process itself by importing the corresponding lib in the ‘run’

- For efficient use of QThread, you should define a worker in a Class that input only (QObject). Then you define your worker in the GUI by passing to init the var you want through the __init__ method, and you simply call worker_whatever.moveToThread(mythread).

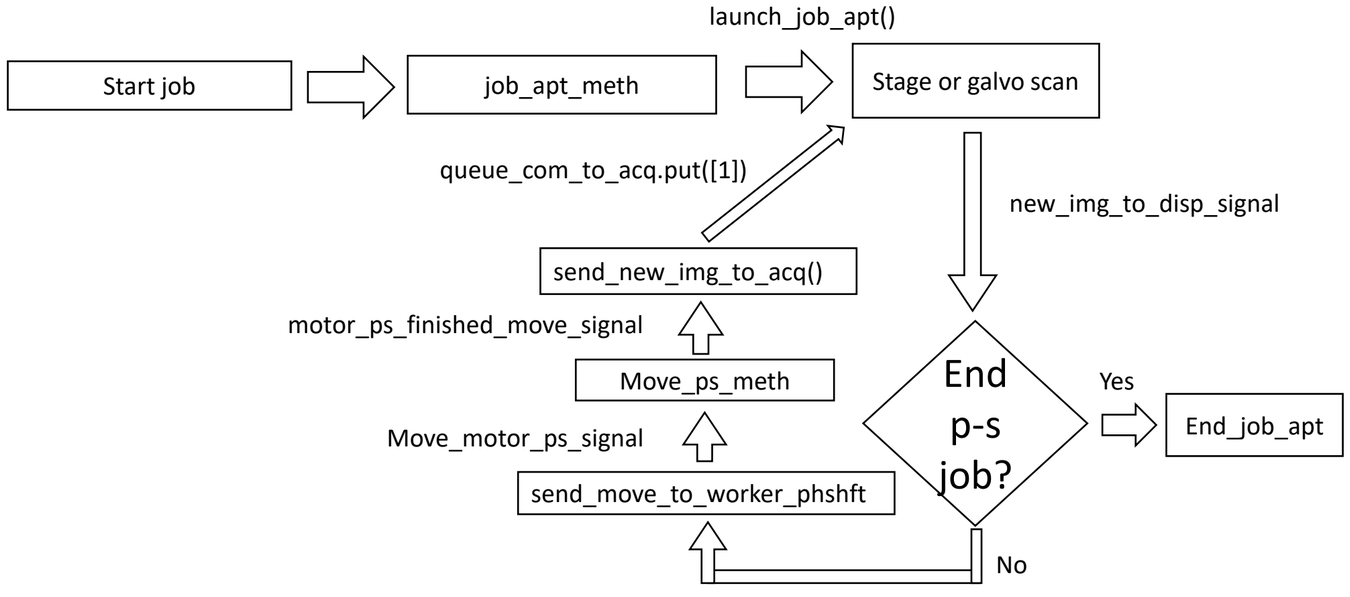
mythread is simply a QThread() that assures that it’s indeed in parallel. You can simply call mythread.start() after to start it.

However, a QThread is not a pure parallel process due to GIL. Use Qprocess or multiprocess instead. But it’s fine to keep the GUI responsive

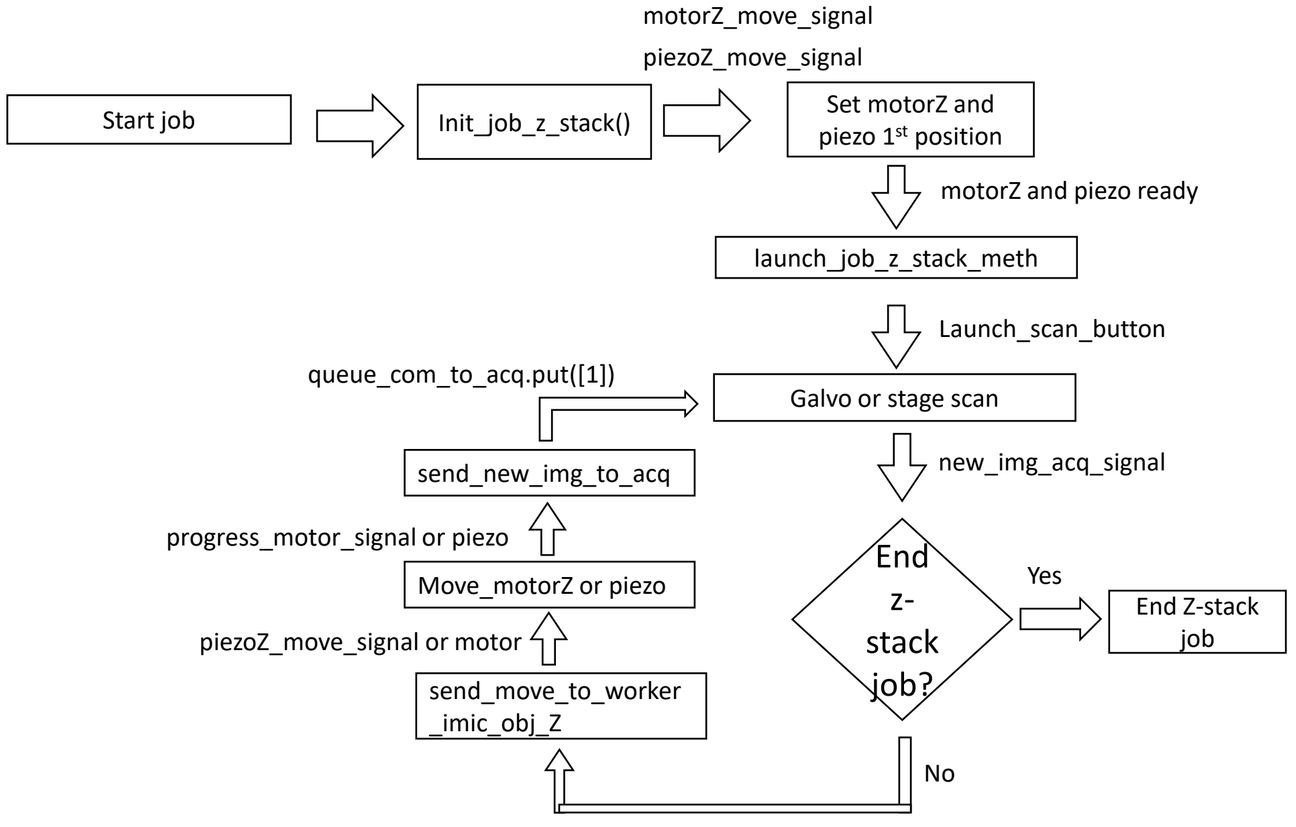

QThread 3
Scan galvos worker
Start
QThread 2
XY APT worker
Start
Connection
Import APT library
QThread 1
Main GUI
Init
setupThread_apt()
setupThread_imic()
Init iMic button pressed
Connection
Home APT XY button pressed
Connection
home_stage_meth
Move motorZ, piezoZ, change objective or bottom filter or top filter action
corresponding_pyQtSign-al.emit(parameters)
QThread 4
iMic worker
Start
Connection
Import iMic library
iMic_ini method
Move motorZ, piezoZ, change objective or bottom filter or top filter method


QThread # 3
APT worker
Move motor rotation
Move motor polarization
Signal
Signal
QThread # 4
iMic
iMic functions : move Z ...
Signals
QThread # 5
PI
PI functions : move Z ...
Signals
Main QThread
Main GUI
APT action
iMic button
QThread # 2
Stage XY
Stage action : move ...
Signal
Move stage
GUI buttons action
Scan stage
Turn on spectro
Signal
Stage scan
Scan galvos
Toggle shutter
Queue for orders
spectrum
Signals
Display
Signal
Queue for stop
Signal
QThread # 6
Phase modulator
Start mode (DC or scan)
Set high voltage
Read status
QThread # 9
Scan galvos
Acquire image
QThread # 8
Shutter worker
Toggle shutter
QThread # 7
Spectro worker
Init instr.
Infinite loop to acquire data (listen for stop)

The jobs, **Case 01 : phase-shifts**

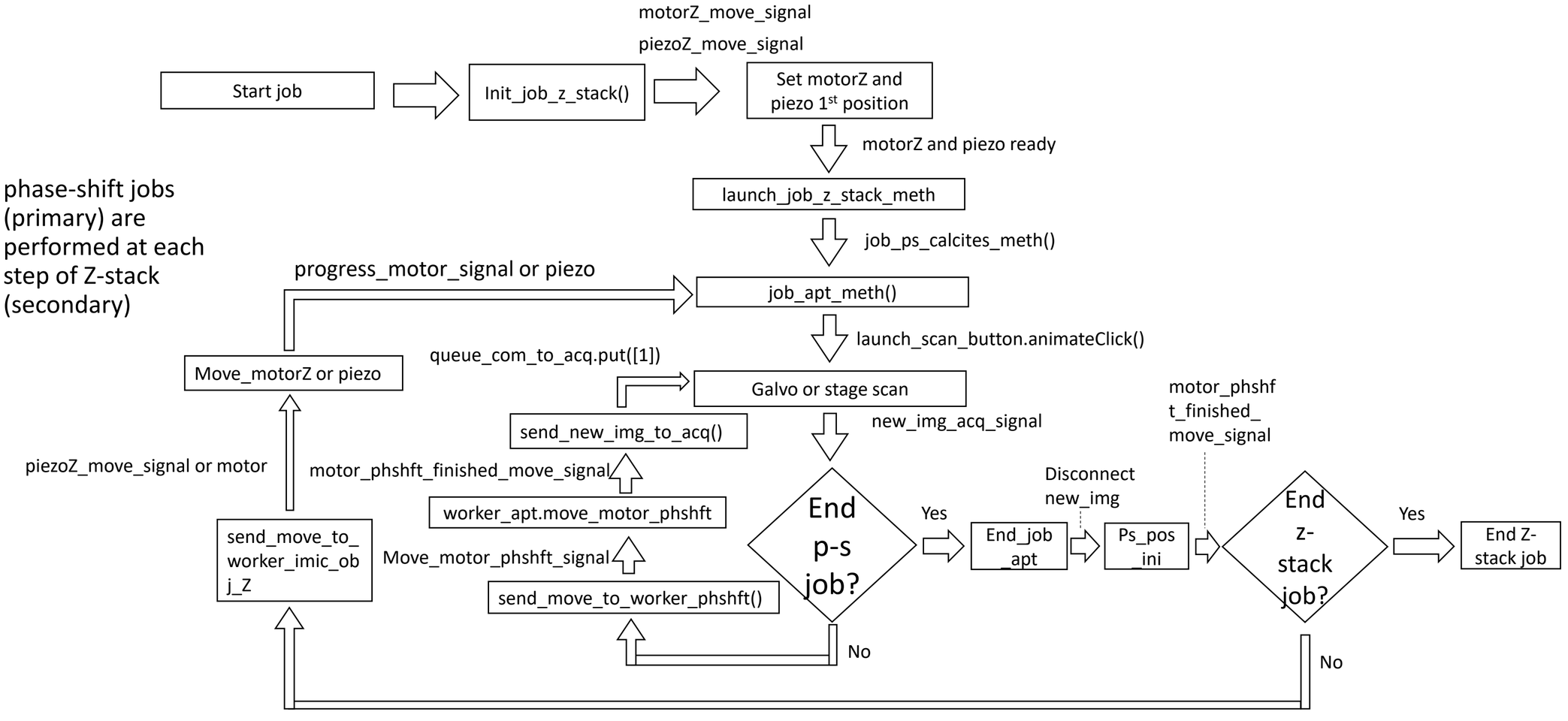


The jobs, **Case 02 : z-stack**

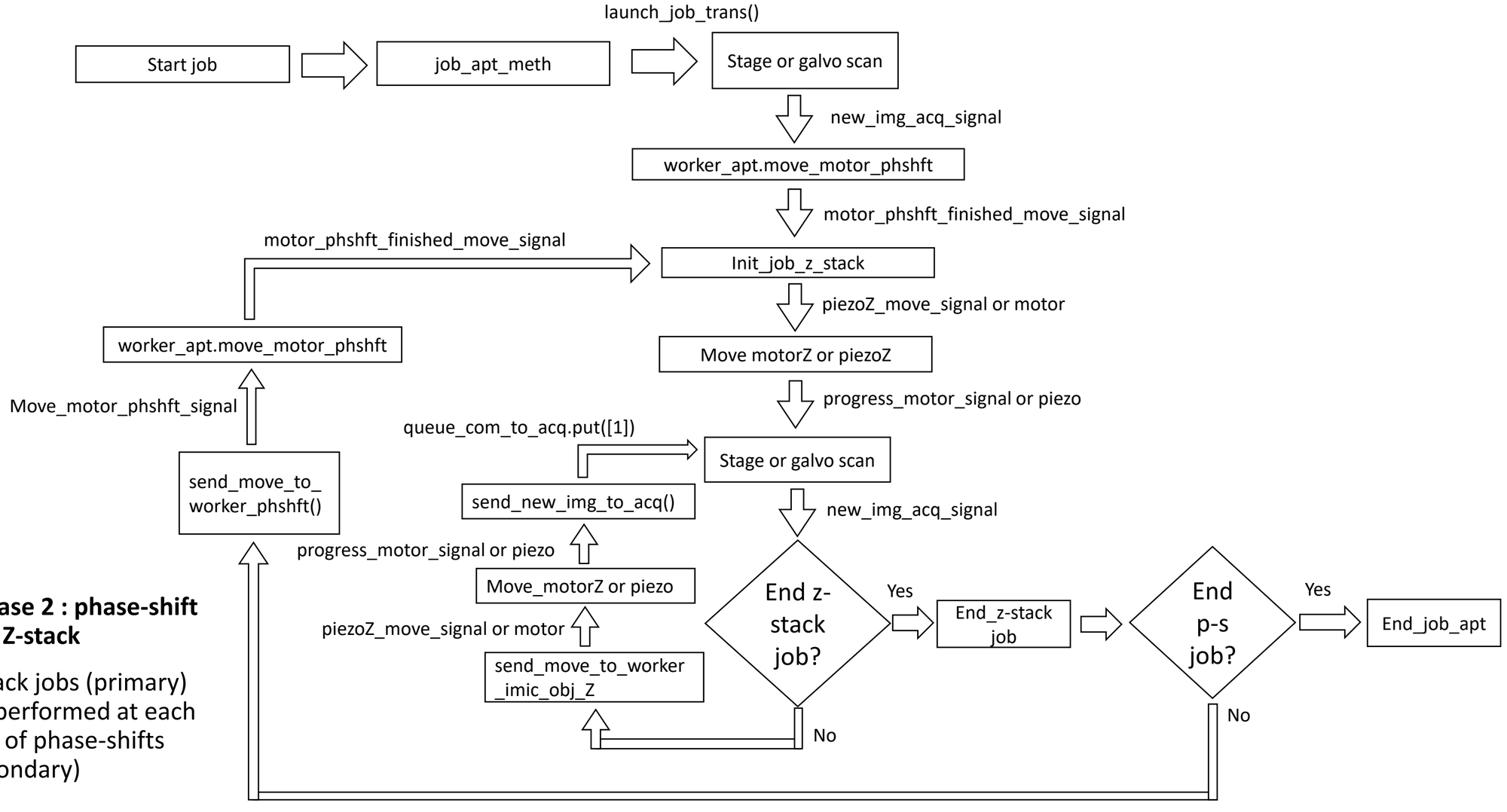


The jobs, **Case 1 : Z-stack > phase-shift (updated 2018/07/09)**

→ phase-shift jobs (primary) are performed at each step of Z-stack (secondary)



The jobs, **Case 2 : phase-shift > Z-stack (updated 2018/07/09)**



- **Case 2 : phase-shift > Z-stack**

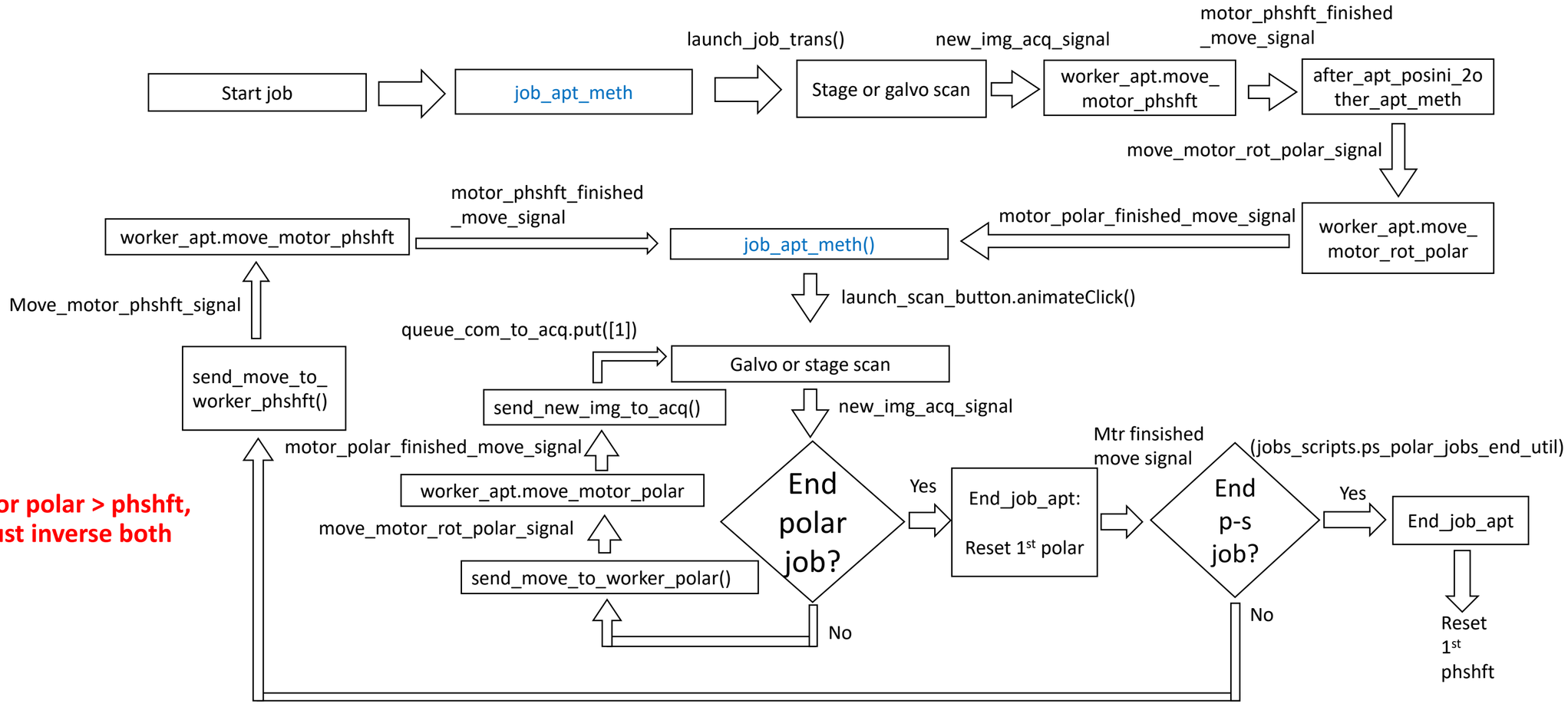
Z-stack jobs (primary) are performed at each step of phase-shifts (secondary)

The jobs, **Case 3 : polar jobs , + Z**

→ Same APT worker

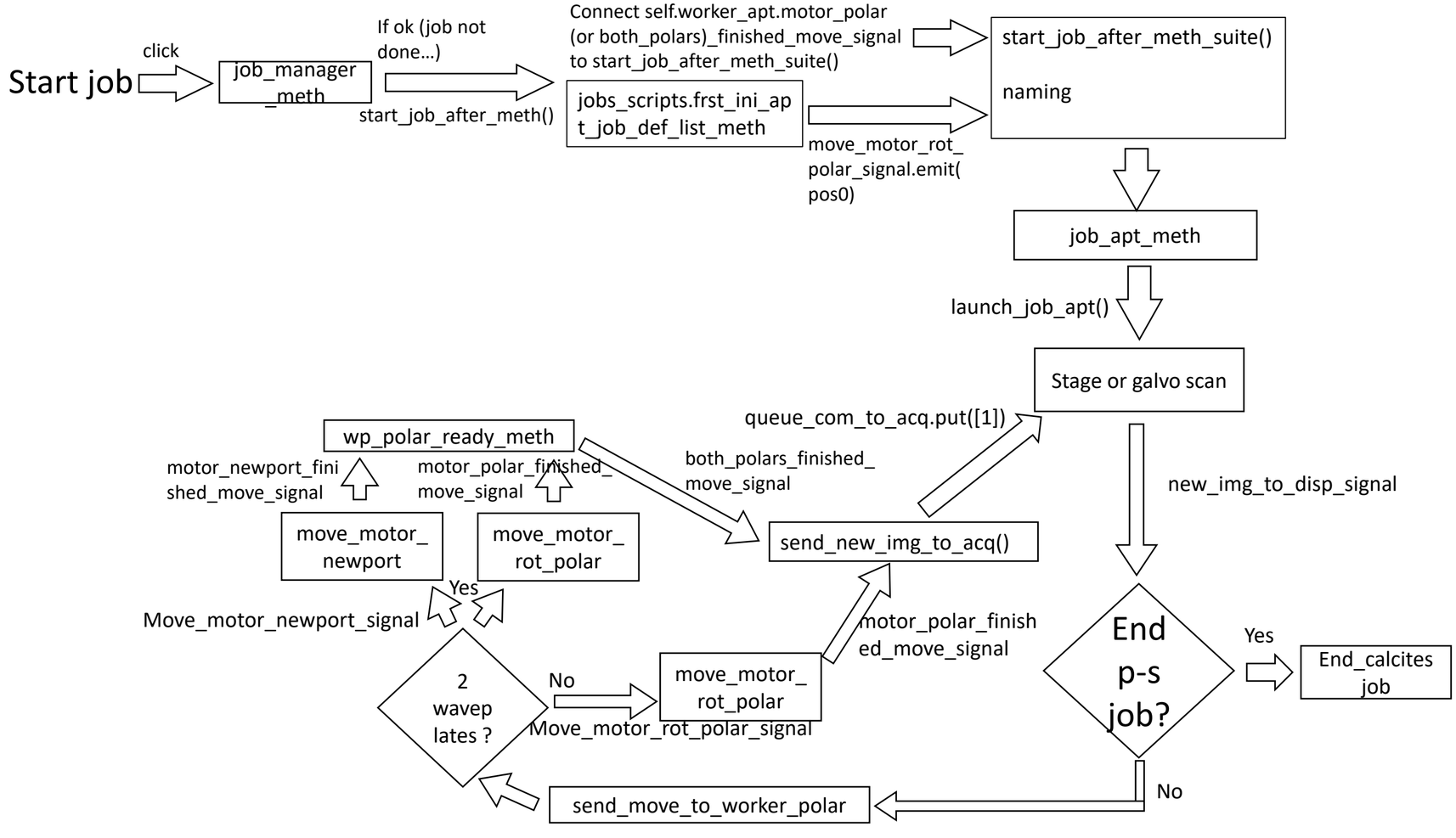
→ Replace phshft by polar

The jobs, **Case 4 : phase-shift > polar (updated 2018/07/09)**

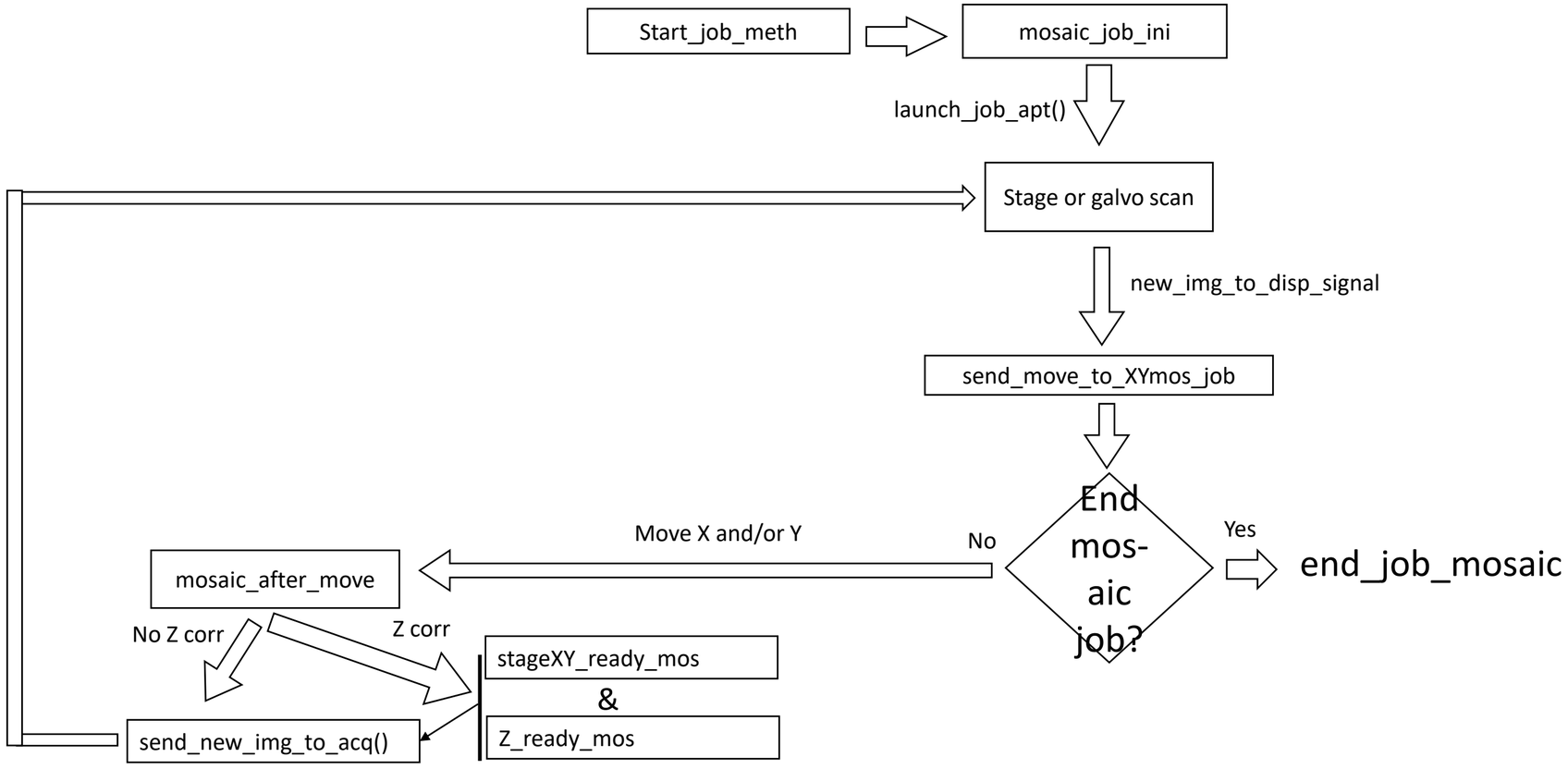


- **For polar > phshft, just inverse both**

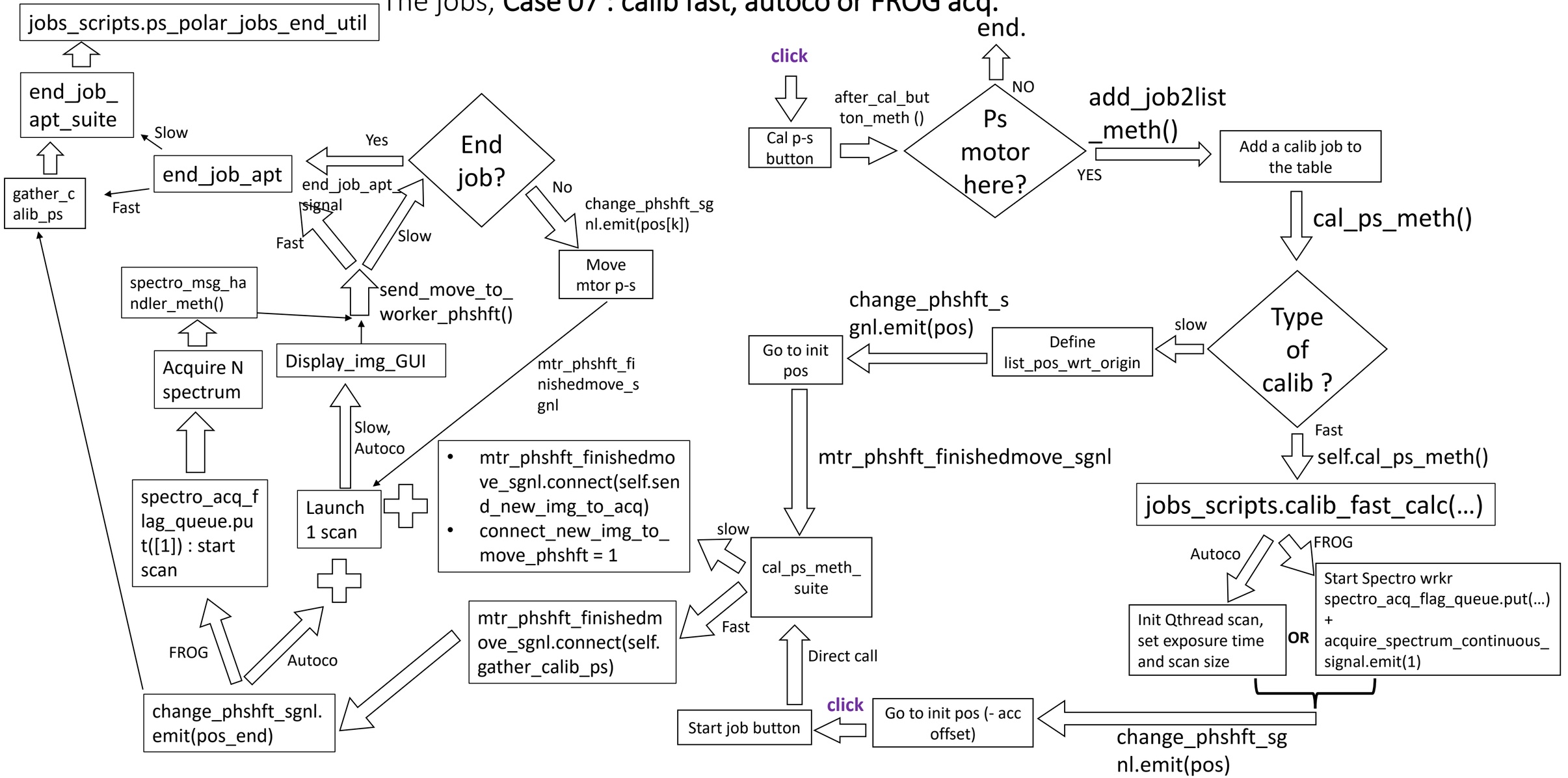
The jobs, **Case 05 : polar with many WP**



The jobs, **Case 06 : mosaic, with or without Z corr.**

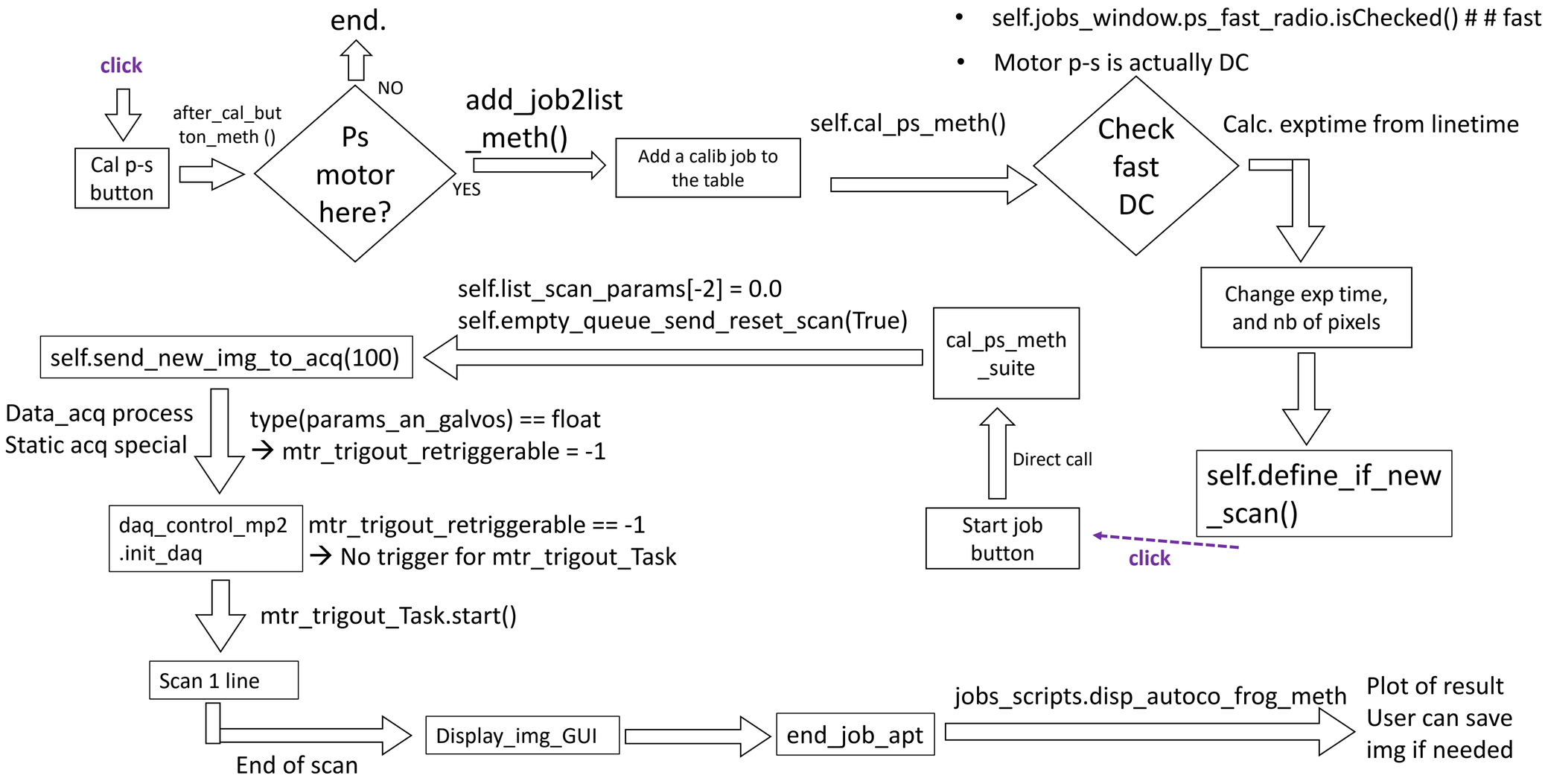


The jobs, **Case 07 : calib fast, autoco or FROG acq.**



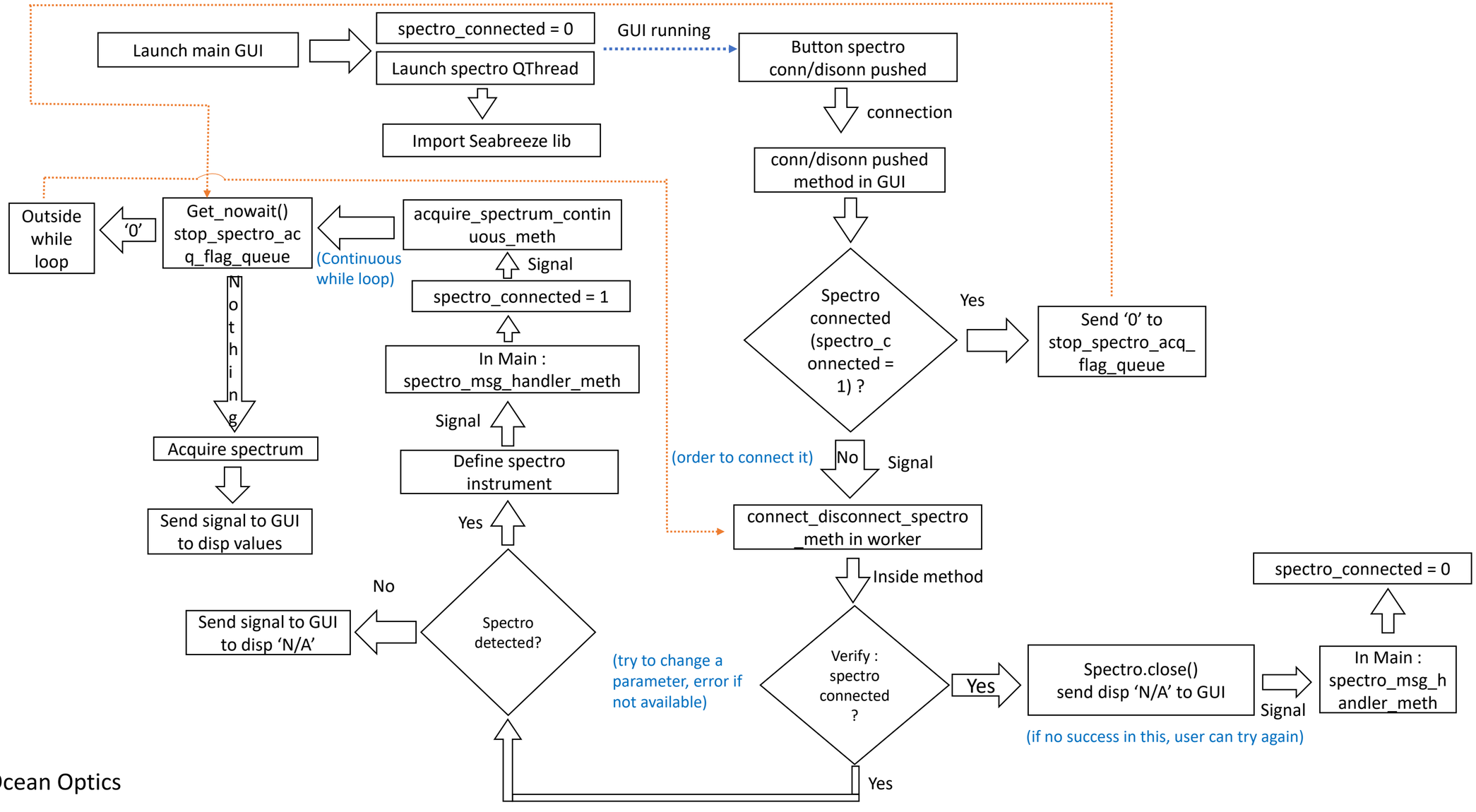
The jobs, **Case 08 : calib fast of EOM AC**

Cond: fast calib mode, with DC as p-s motor (in 2nd job window)
→ User has to put himself the right ramp mode

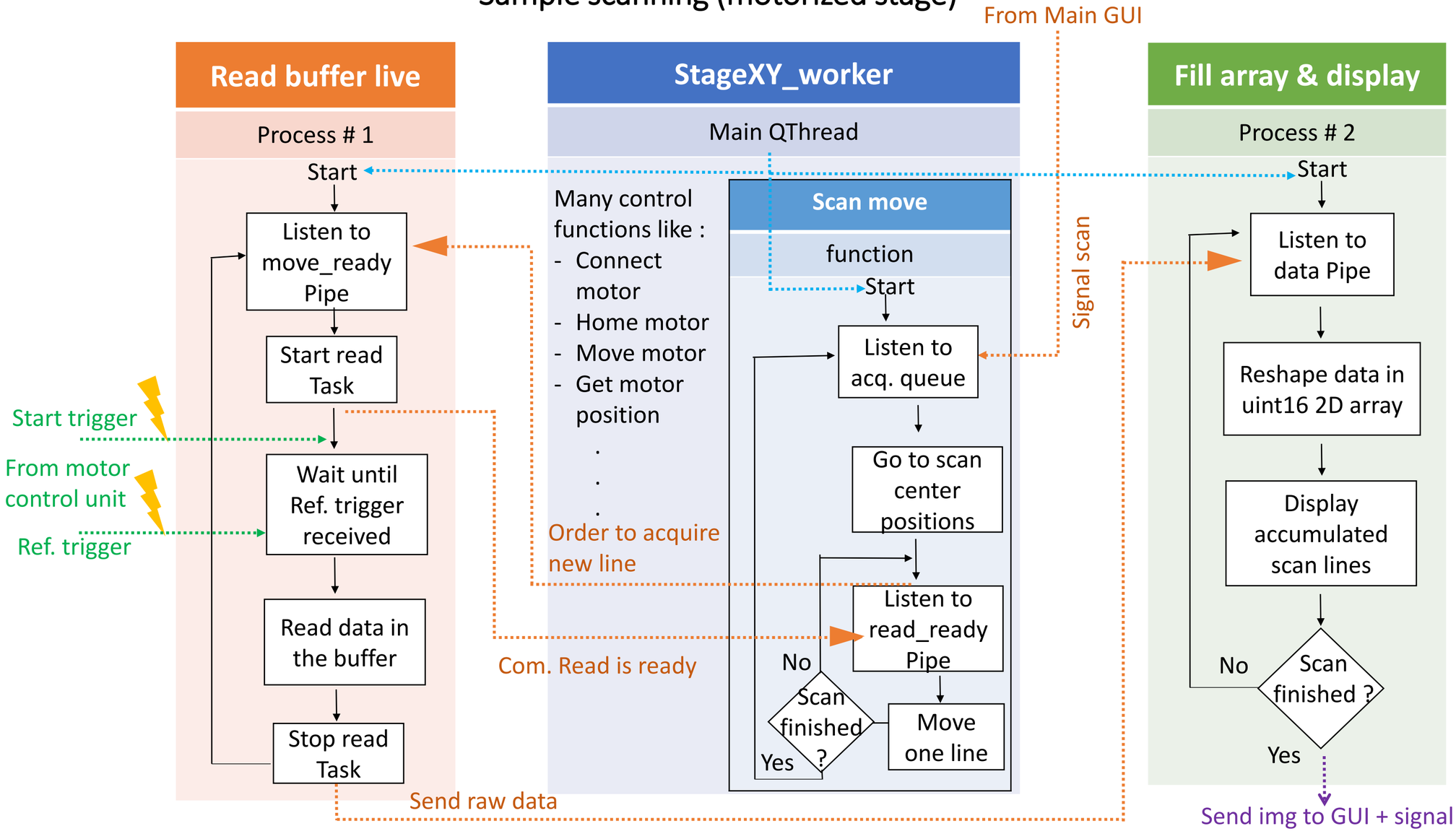


- self.jobs_window.ps_fast_radio.isChecked() # # fast
- Motor p-s is actually DC

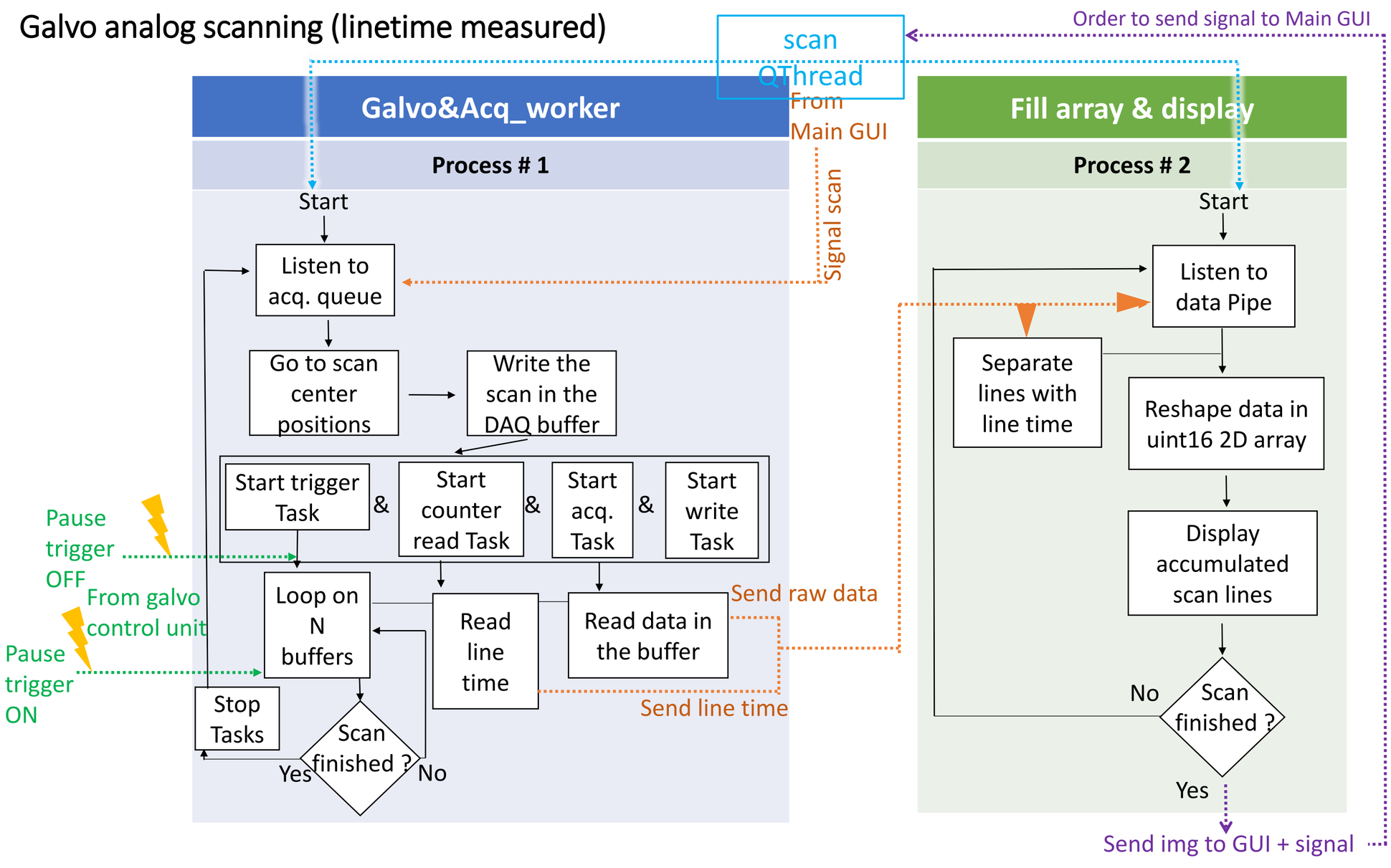
Spectro QThread



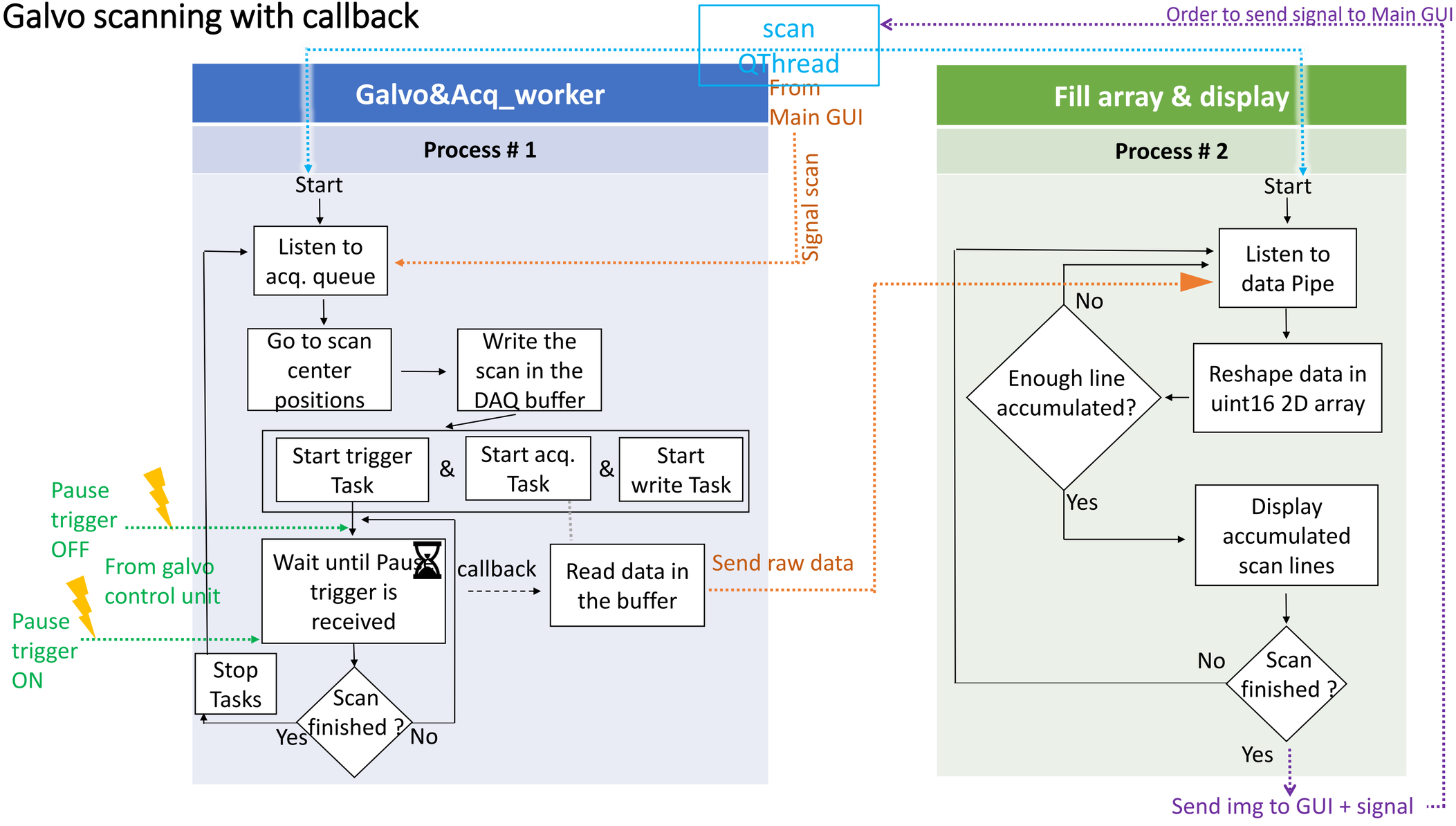
Sample scanning (motorized stage)

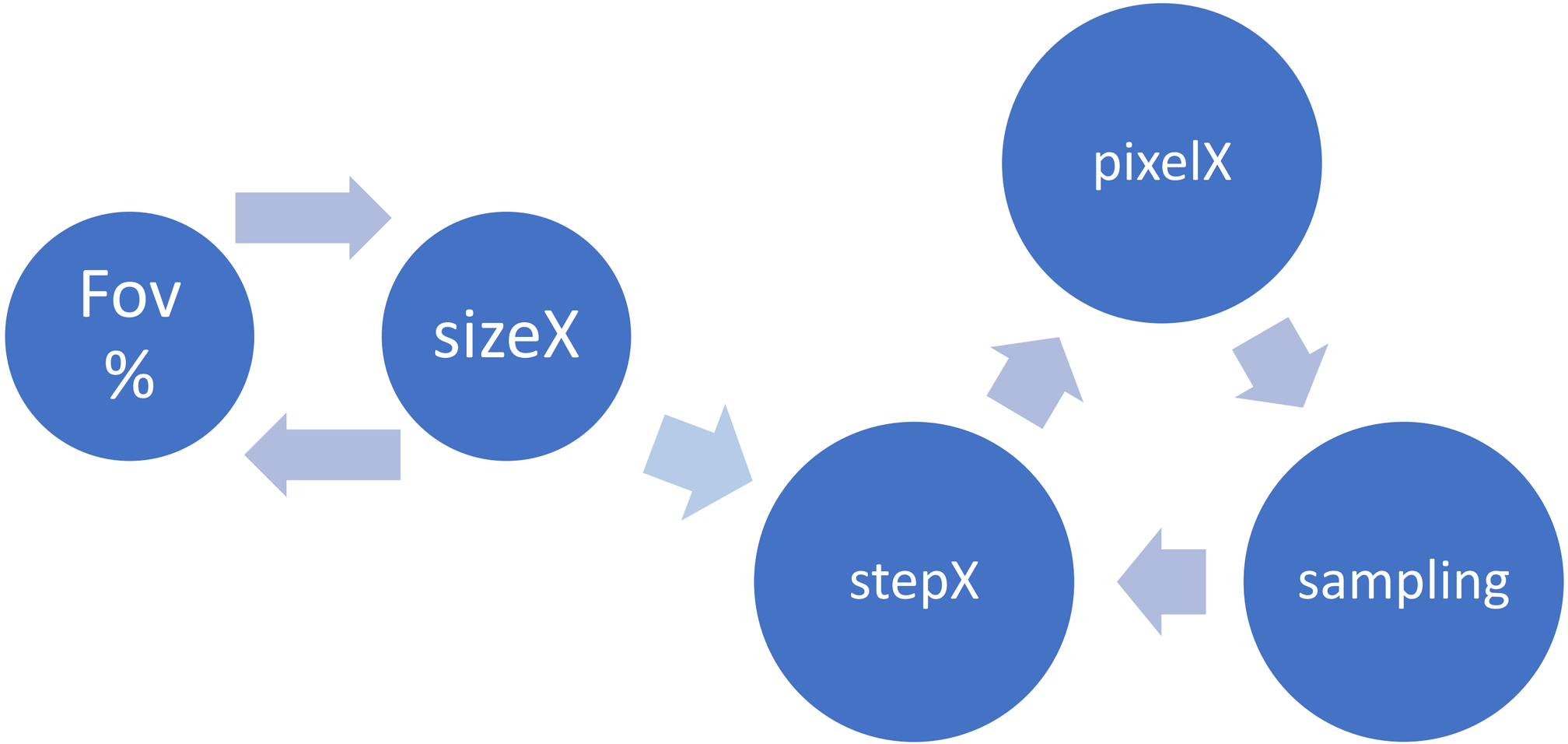


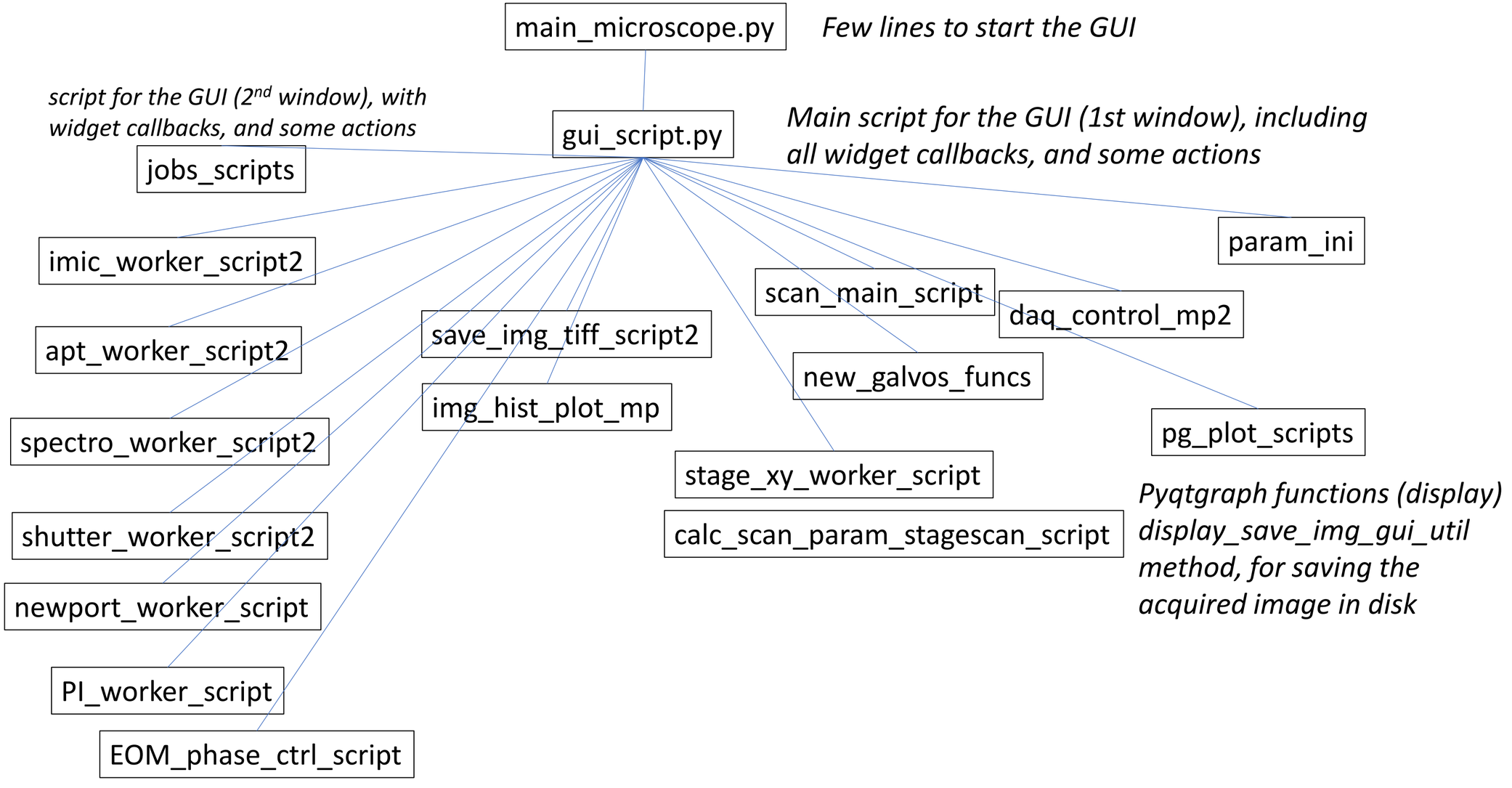
Galvo analog scanning (linetime measured)

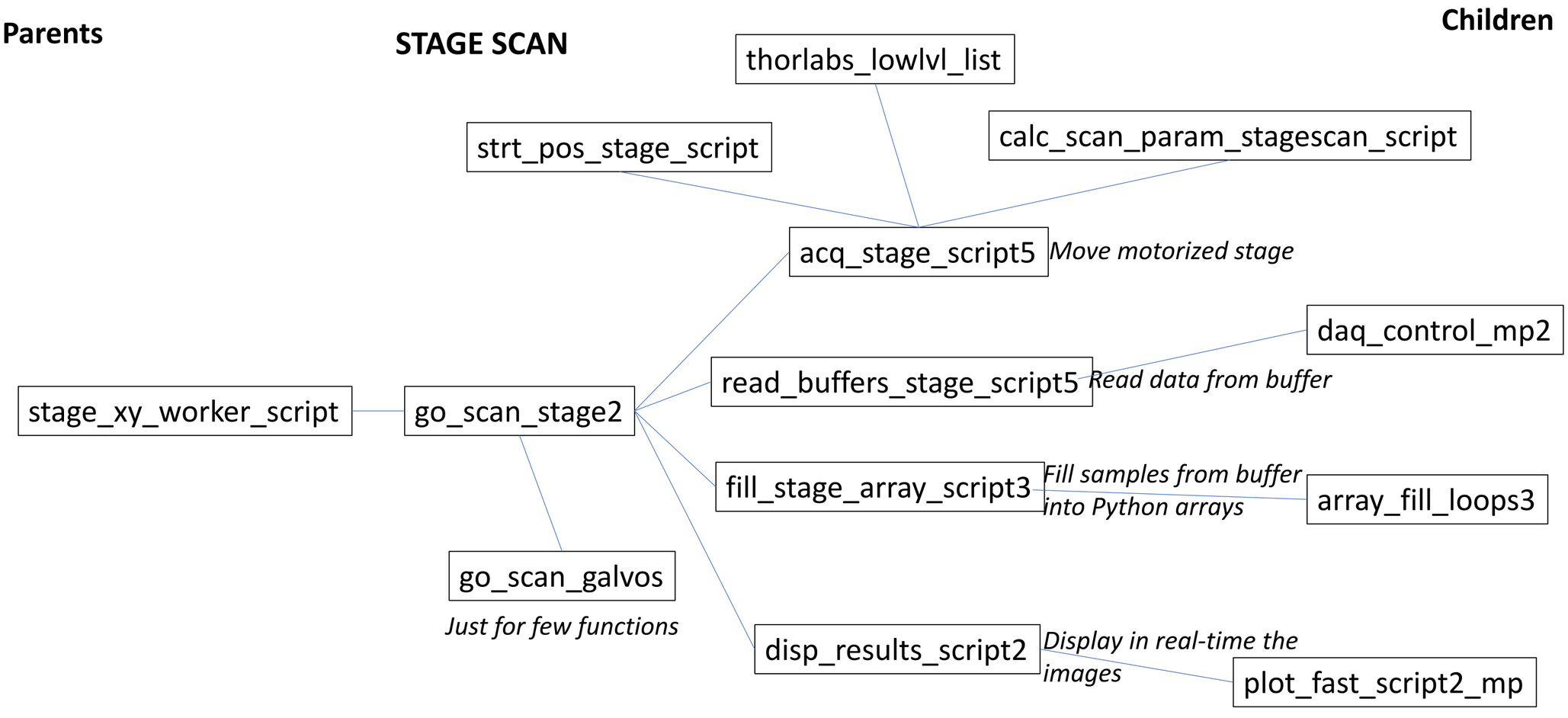


Galvo scanning with callback




Fov
%
sizeX
pixelX
stepX
sampling

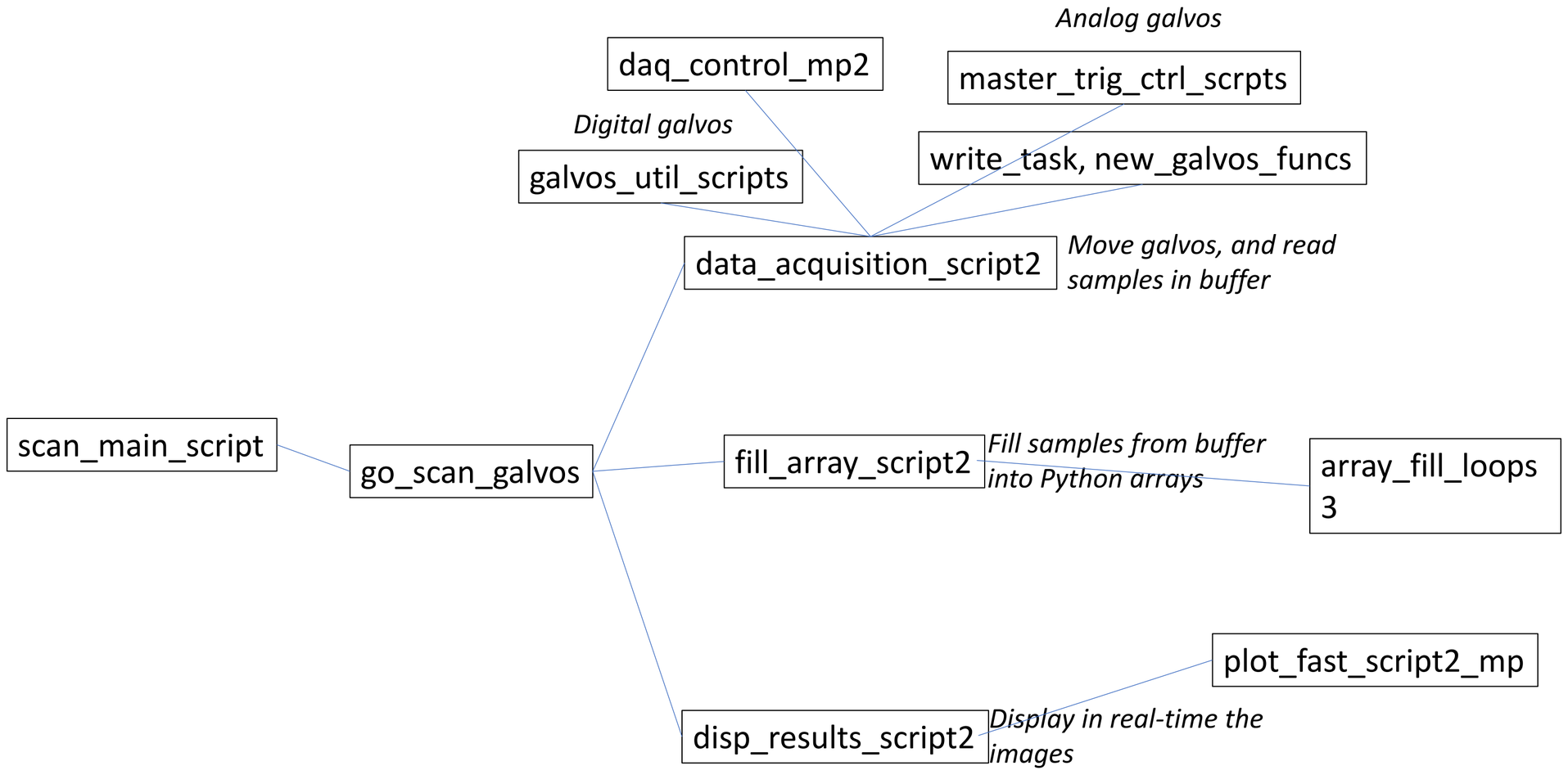

main_microscope.py
Few lines to start the GUI
script for the GUI (2nd window), with widget callbacks, and some actions
gui_script.py
Main script for the GUI (1st window), including all widget callbacks, and some actions
jobs_scripts
imic_worker_script2
apt_worker_script2
spectro_worker_script2
shutter_worker_script2
newport_worker_script
PI_worker_script
EOM_phase_ctrl_script
save_img_tiff_script2
img_hist_plot_mp
param_ini
scan_main_script
daq_control_mp2
new_galvos_funcs
pg_plot_scripts
stage_xy_worker_script
calc_scan_param_stagescan_script
Pyqtgraph functions (display)
display_save_img_gui_util method, for saving the acquired image in disk


Parents
STAGE SCAN
Children
thorlabs_lowlvl_list
strt_pos_stage_script
calc_scan_param_stagescan_script
acq_stage_script5
Move motorized stage
daq_control_mp2
read_buffers_stage_script5
Read data from buffer
stage_xy_worker_script
go_scan_stage2
fill_stage_array_script3
Fill samples from buffer into Python arrays
array_fill_loops3
go_scan_galvos
Just for few functions
disp_results_script2
Display in real-time the images
plot_fast_script2_mp

Parents

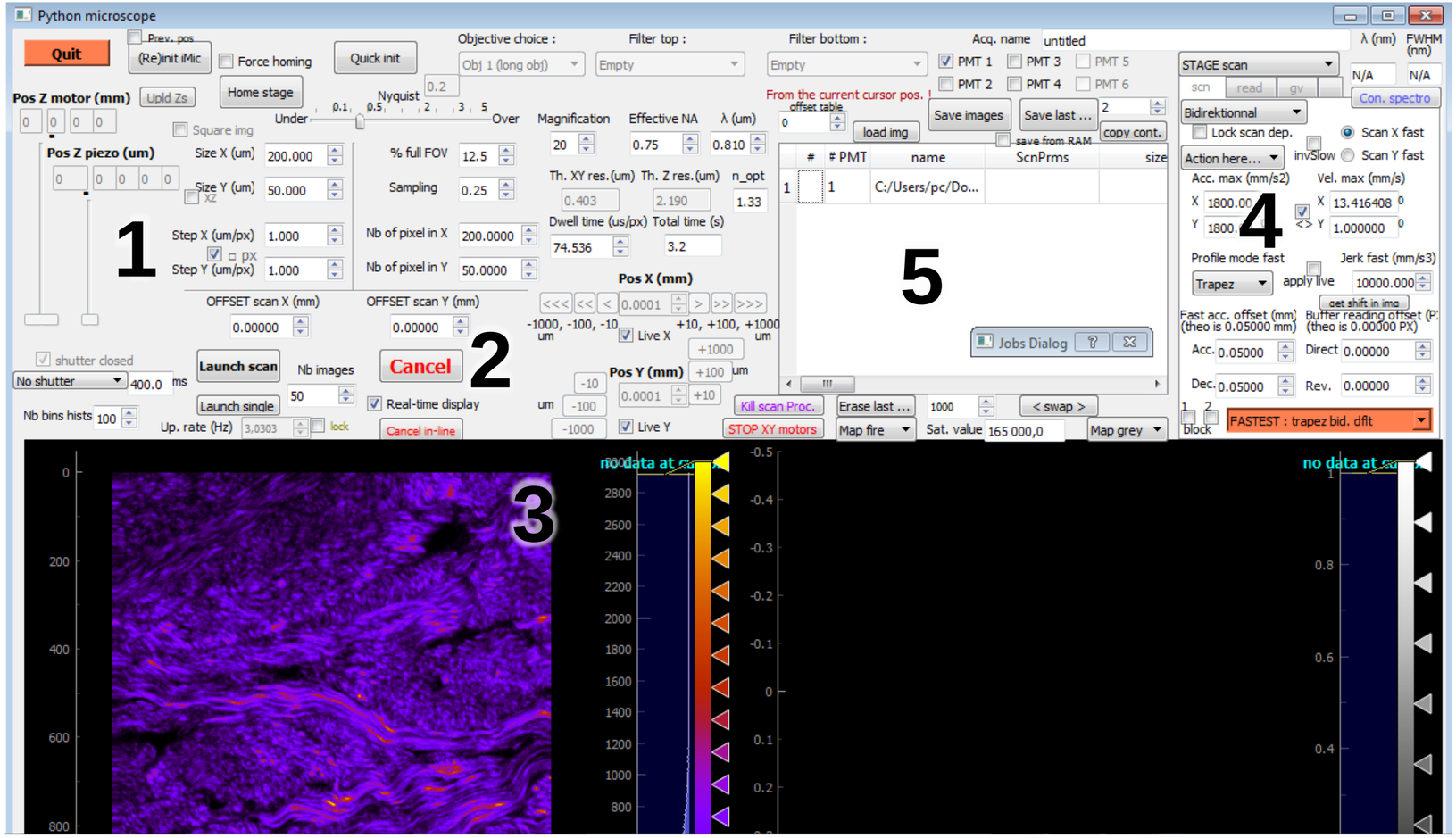
GALVOS SCAN

Children

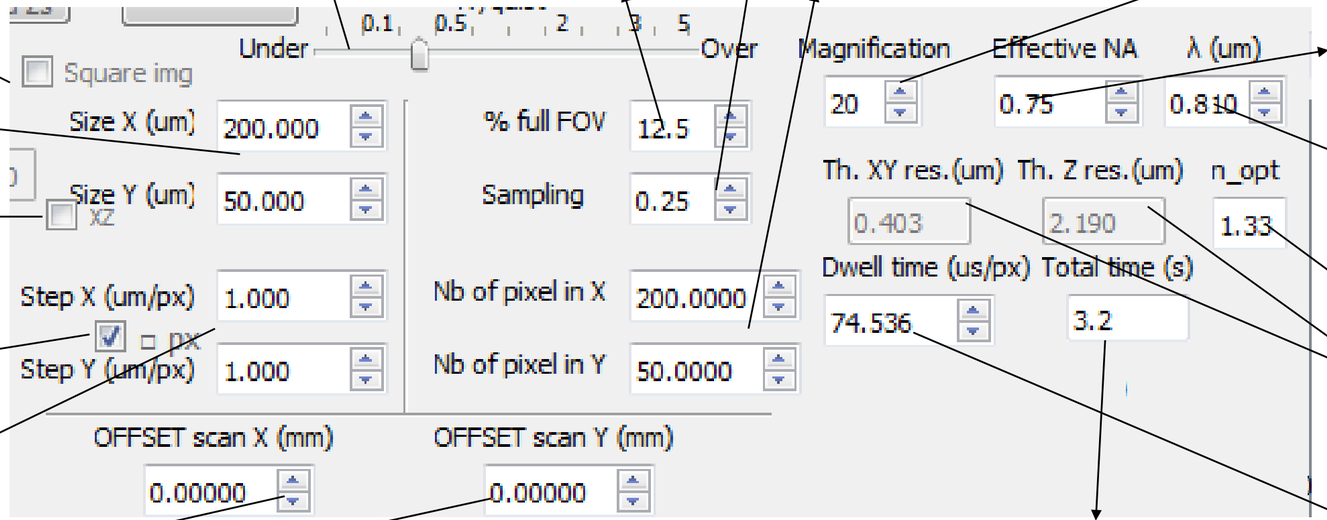


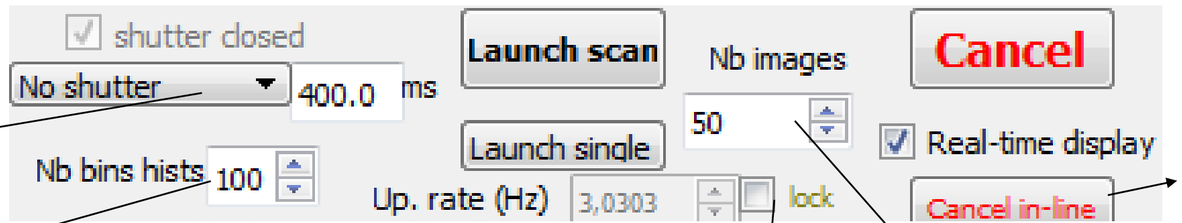
Full GUI description

Full GUI description in Python



1) GUI : (main) scan parameters





A scan step corresponds to a sampling

A scan size corresponds to a % of the FOV

Ratio to the Nyquist factor

Corresponding number of pixels, knowing the scan size and step

Magnification of the objective

Impose square image

N.A. (measured) of objective

SizeX/Y in µm

Fundamental wavelength

Do a XZ scan instead of XY (need modif to be implemented)

Index of medium (sample)

Impose square pixels

THEORETICAL resolution in SHG (Zipfel et al.)

Step X/Y in µm/px

Exposure time/px

Total time of img (s)

Offset in X/Y of the scan, used with galvos to change the 0 pos of the alignment.

Use of shutter, or not

Nb of bins for the 2D histogram

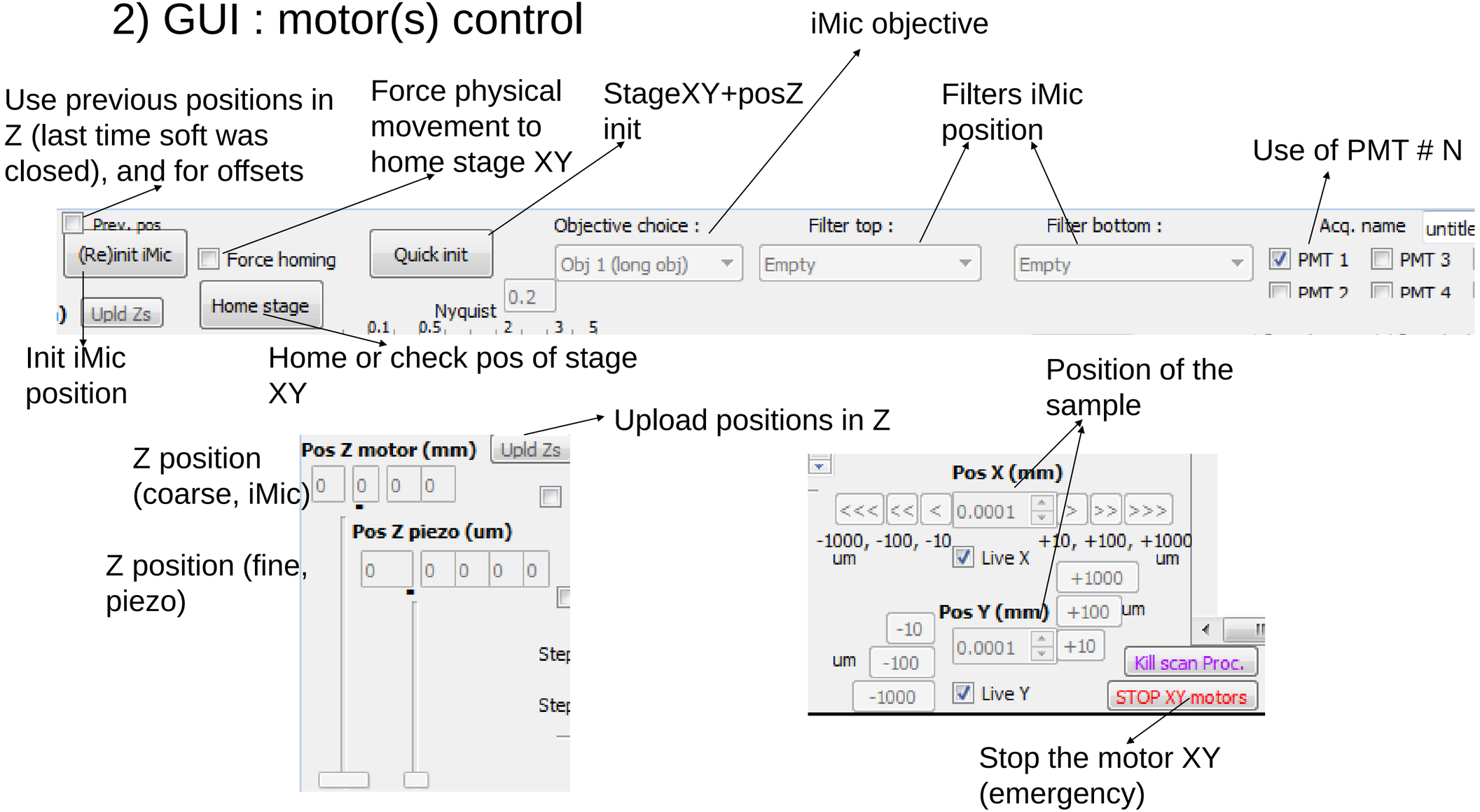
Update rate of buffers (for static acq., or galvos)
Higher rate→ smaller packets

Lock update rate

Nb of images to acquire

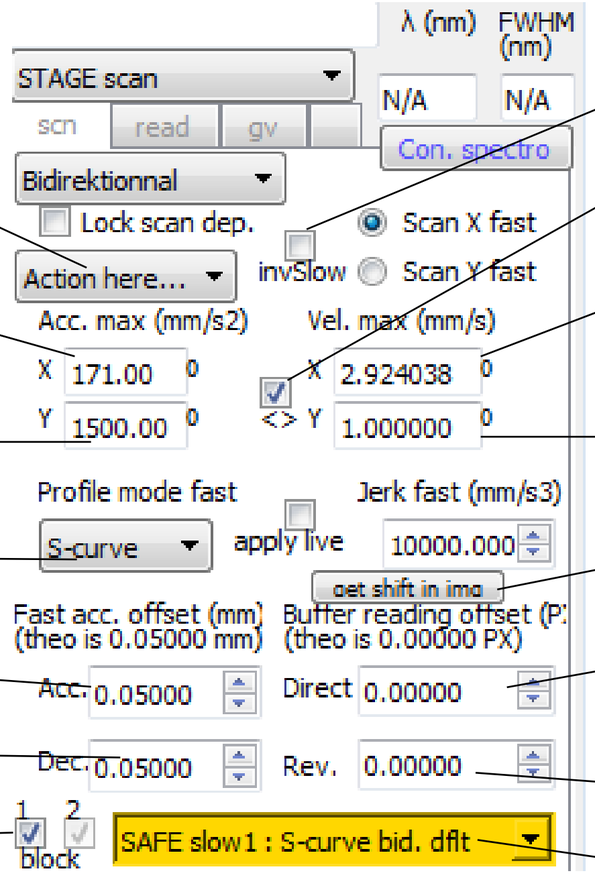
Cancel image during acquisition (not ok for dig galvos, as scan has to finish anyways)

2) GUI : motor(s) control



4) GUI: Detailed parameters of the mode

FOR STAGE SCAN



Force update, or set of optimal params, or lock params

Acceleration of the motor (X, usually fast)

Acceleration of the motor (Y, usually slow)

Scan mode profile

Acceleration offset (spatial)

Deceleration offset (spatial)

Block until end of move (can uncheck for fast)
See stage_scan summary

Invert slow direction

Enable/disable link between acceleration and speed of motor

Velocity of the motor (X, usually fast)

Velocity of the motor (Y, usually slow)

Get shifts (fast unprecise, use DIPimage for more precise)

reading offset (in buffer)

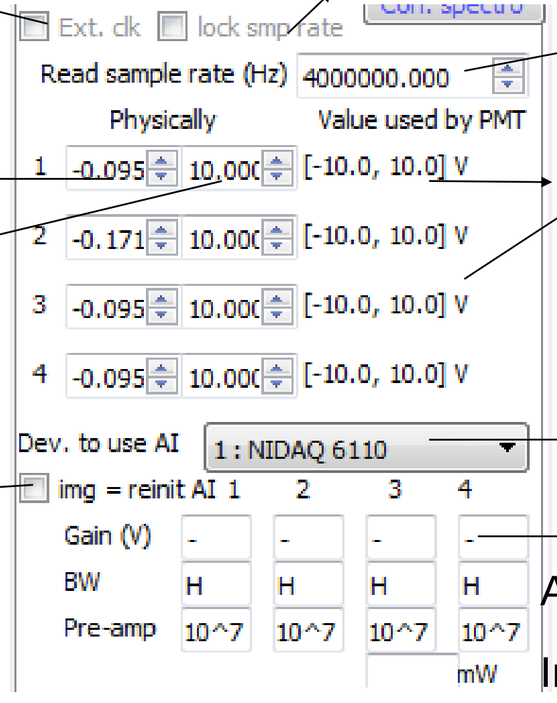
reading offset (in buffer, reverse)

Easy change of mode

4b) GUI: Detailed parameters of the mode

Use external clock for sample rate

Lock sample rate, to avoid it to vary with other parameters



Sample rate for acquisition

Min and max values physically given by the PMT

Value used by daq card for analog input bounds, that determines the precision (for PMT 1, 2, …)

NI card to use for acquisition (2 choices)

Check to re-init Tasks of card at each images

Gain on amplifier of PMT

Anode amplifier carac

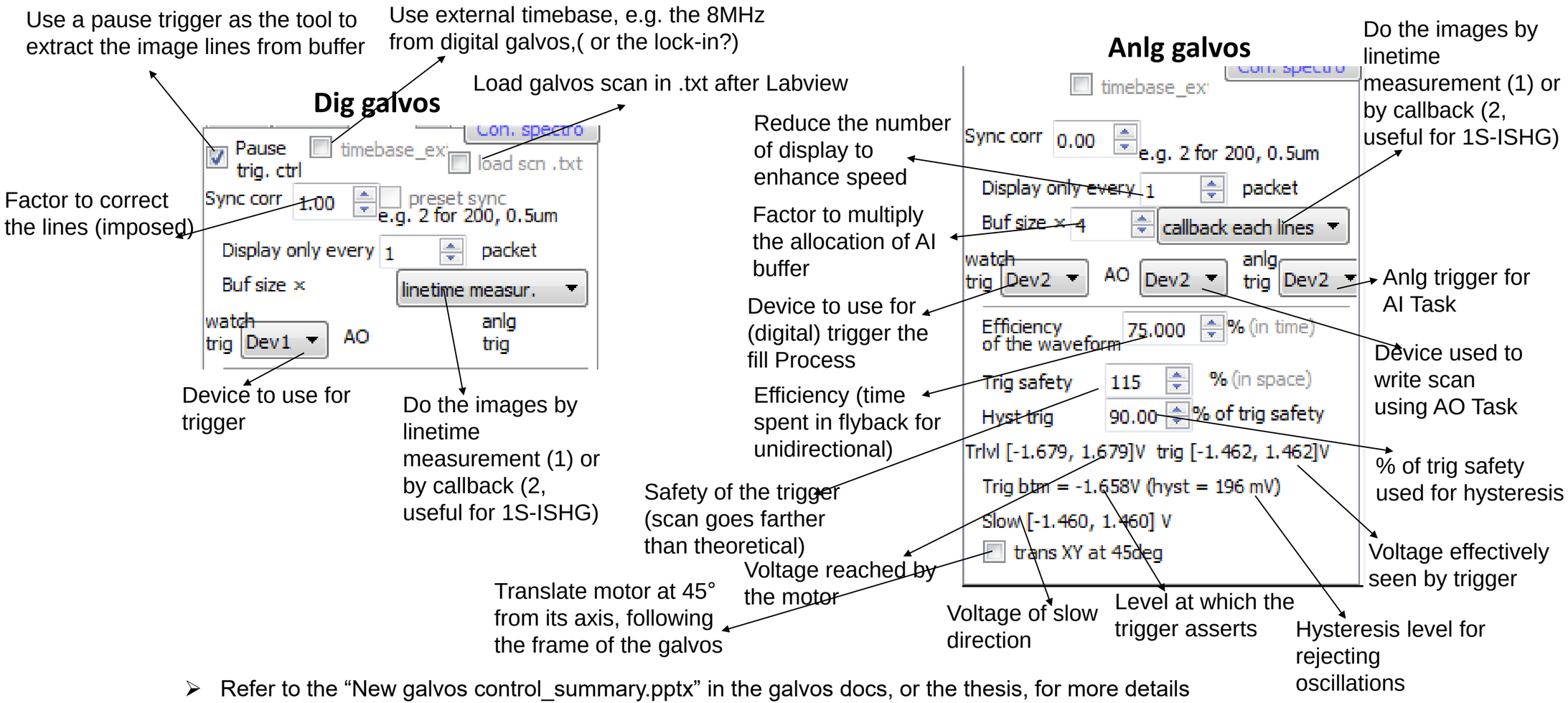
Input optical power

LAST TAB

Draw a rectangle on current image to define a ROI

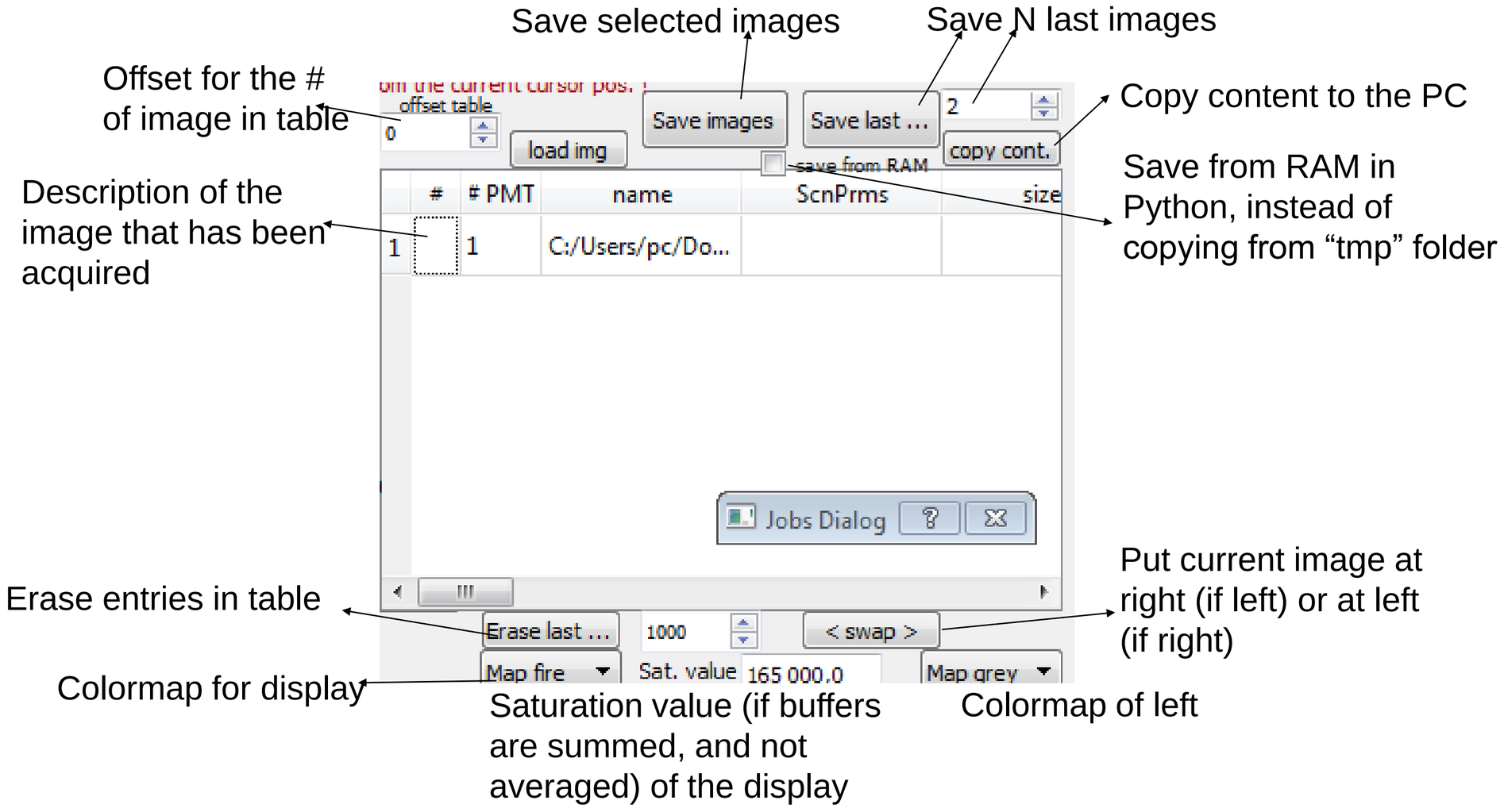
Define the drawn ROI in scan parameters

4c) Galvos, anlg & dig

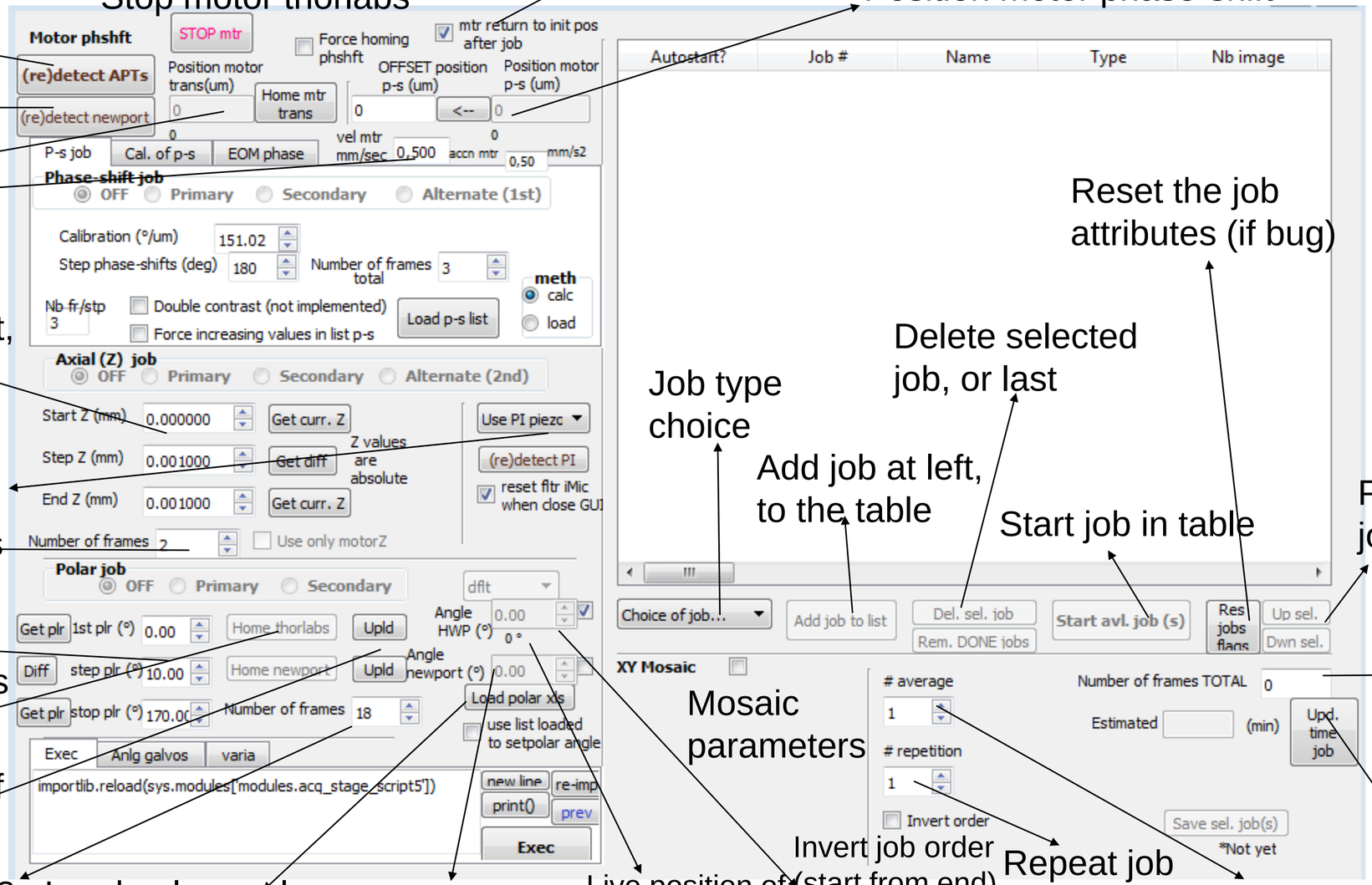


- Refer to the “New galvos control_summary.pptx” in the galvos docs, or the thesis, for more details

5) GUI: image display



6) GUI: jobs



Stop motor thorlabs

return to init position after job of phase-shift

Position motor phase-shift

Detect Thorlabs cube motors

Detect Newport motor

Motor translation

Velociy/accn of motor phase-shift

Reset the job attributes (if bug)

Z motor job: start, step and stop positions

Delete selected job, or last

Job type choice

Add job at left, to the table

Start job in table

Put selected job up or down

Detect PI piezo (Z)

Number of frames for Z jobs

Polar start, step, and stop positions

estimation of job duration (automatic normally)

Mosaic parameters

Upload position of waveplates

Upload estimation of job duration (automatic normally)

Nb of frames of polar job

Load polar angles, HWP+QWP

Position of QWP (Newport)

Live position of HWP

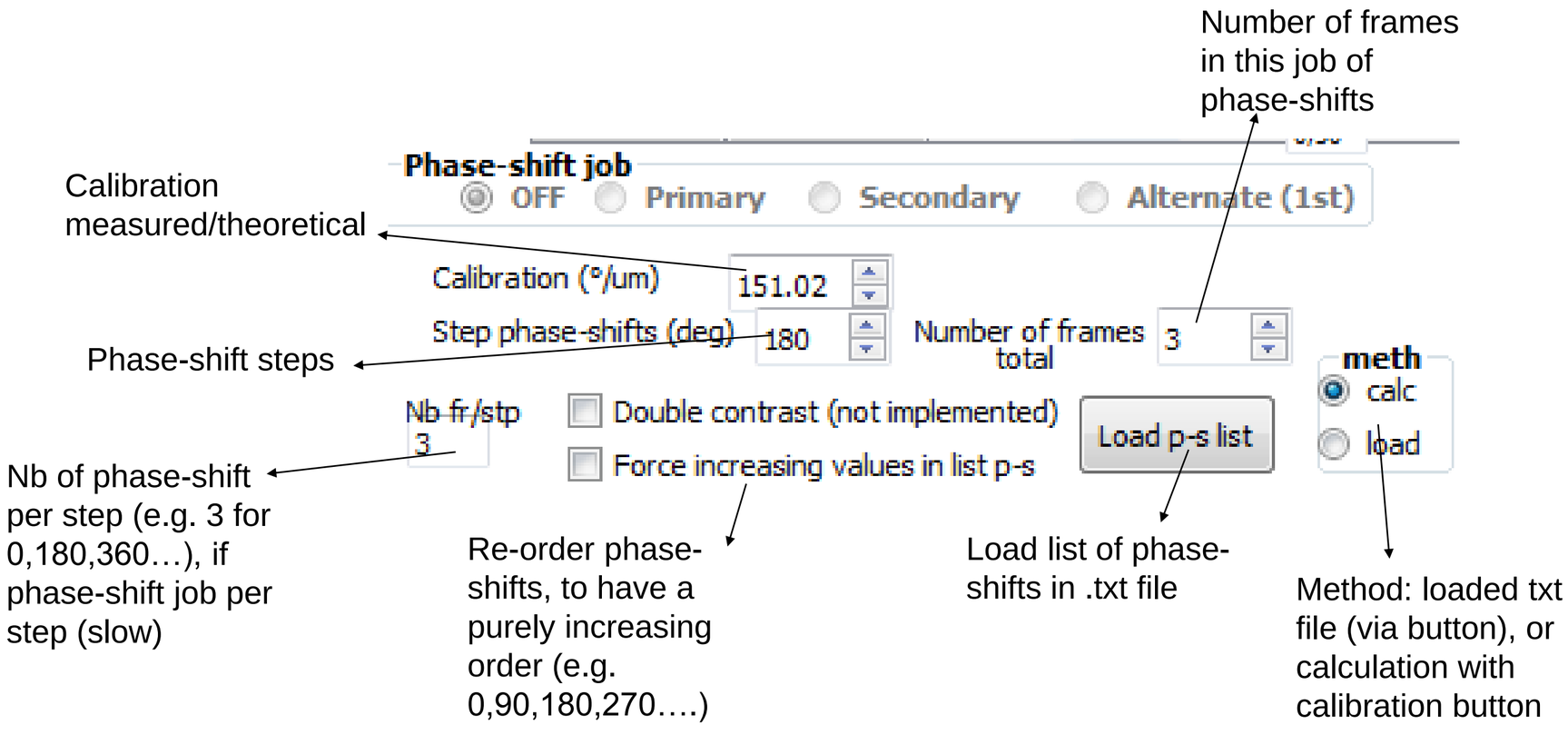
Change HWP (Thorlabs) position

Invert job order (start from end)

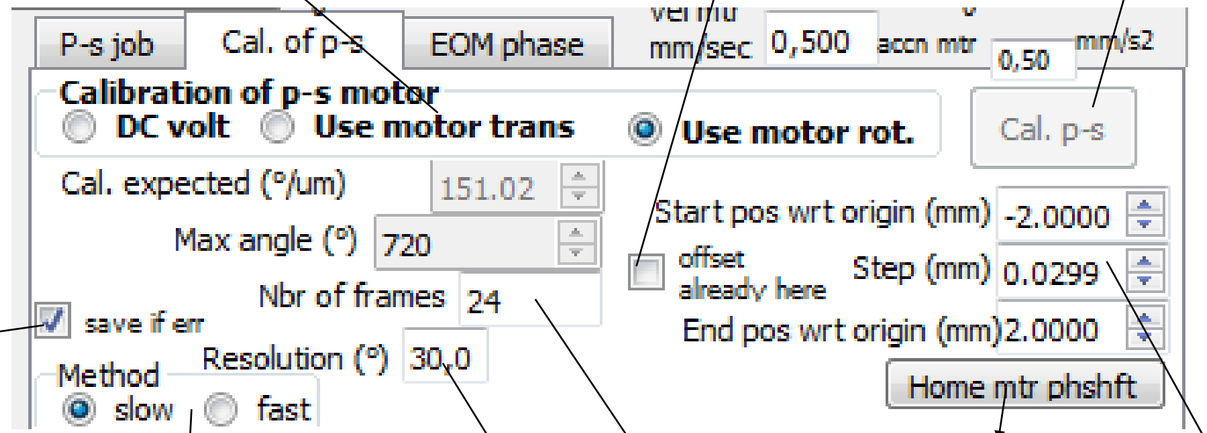
Repeat job N times

Number of average in jobs

6a) GUI: jobs


Number of frames in this job of phase-shifts
Calibration measured/theoretical
Phase-shift job
OFF
Primary
Secondary
Alternate (1st)
Calibration (°/um)
151.02
Step phase-shifts (deg)
180
Number of frames total
3
Phase-shift steps
meth
calc
load
Nb fr/stp
3
Double contrast (not implemented)
Force increasing values in list p-s
Load p-s list
Nb of phase-shift per step (e.g. 3 for 0,180,360…), if phase-shift job per step (slow)
Re-order phase-shifts, to have a purely increasing order (e.g. 0,90,180,270….)
Load list of phase-shifts in .txt file
Method: loaded txt file (via button), or calculation with calibration button

6b) GUI: jobs



Motor for phase-shift, or calib

If checked, don't translate motor to compensate acceleration for calib

Start calib

Save calib if error during acquisition

Calib slow (step-by-step), or fast (one translation)

Resolution of calib

Number of frames in calib

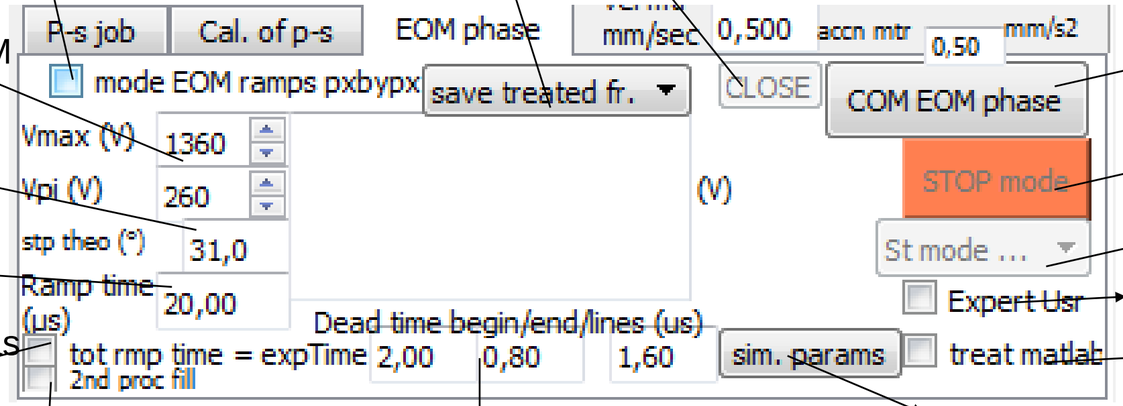
Home motor phase-shift (optional)

Start/step/stop positions for calib

Save only treated interferograms, or whole arrays, or both

Acquire with EOM phase treatment (even if no mode is started

Close EOM



Voltages of EOM (Vpi and max)

Step of phase-shift

Ramp time of EOM mode

Impose EOM mode as pixel exposure time

Start a 2nd Process to fill buffer for EOM

Dead-times of mode EOM

Connect phase EOM

Stop EOM mode

Start EOM mode

Show advanced buttons of EOM

Treat ps job in MatLab

Display parameters of EOM scan

6c) GUI: jobs

Zone to input text, to be executed in Python's console in live (to modify code, or reload a script)

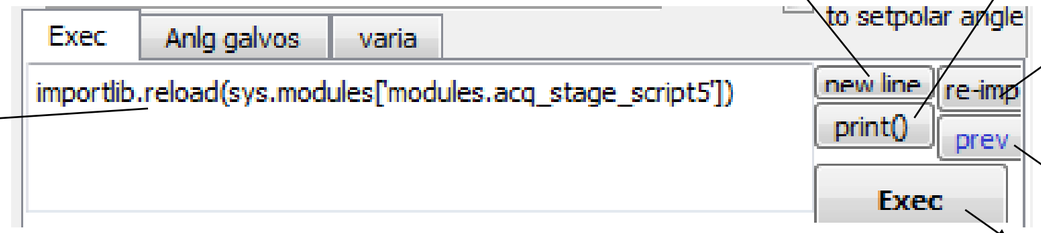
Break a line in the plain Text

Put print balise in widget's text

Put the text to reimport module in PlainText

Display previous commands

execute command in PlainText

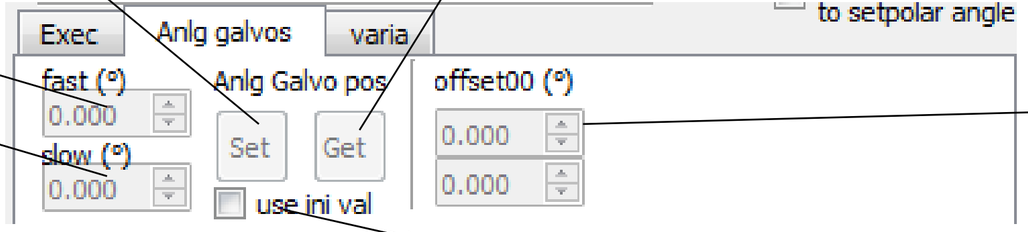


Set positions of widget

Get positions of galvos, by plugging cables

Fast/slow position of anlg galvos, to set

Offset of anlg glavos



Check to use values saved in txt file (previous)

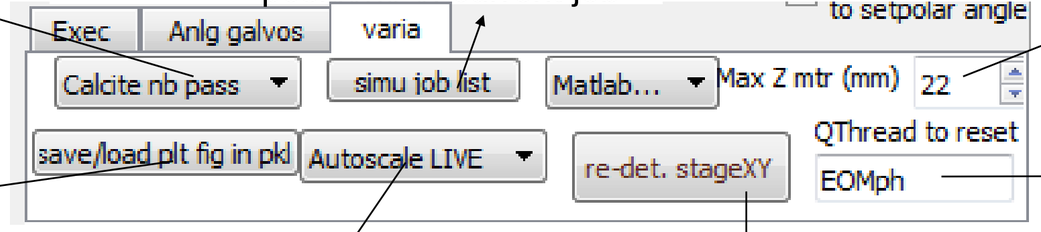
For calib autoco, number of pass in calcite (for calculations)

Display (in logger) the list of positions of current job

maximum value of Z for objective (can limit to a smaller value to avoid hitting the sample)

Save plotted figure in pkl (left-click), or load a .pkl file (right-click)

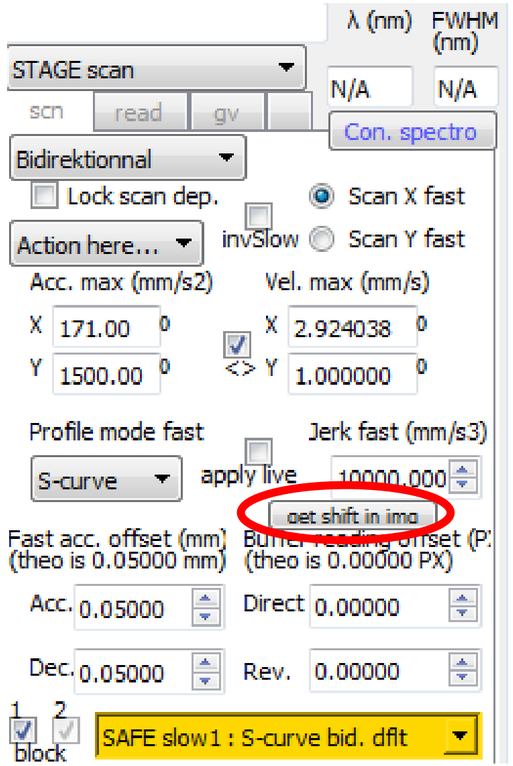
By pressing 'Enter', put widget's text in PlainText for reloading a QThread: shutter, apt, …



Autoscale LIVE, or 90% of dynamic range, or 1st buffer

(re)detect stage Thorlabs XY (if was closed for bug)

Get accurate shift lines in bidirek stage scan



DIPIMAGE

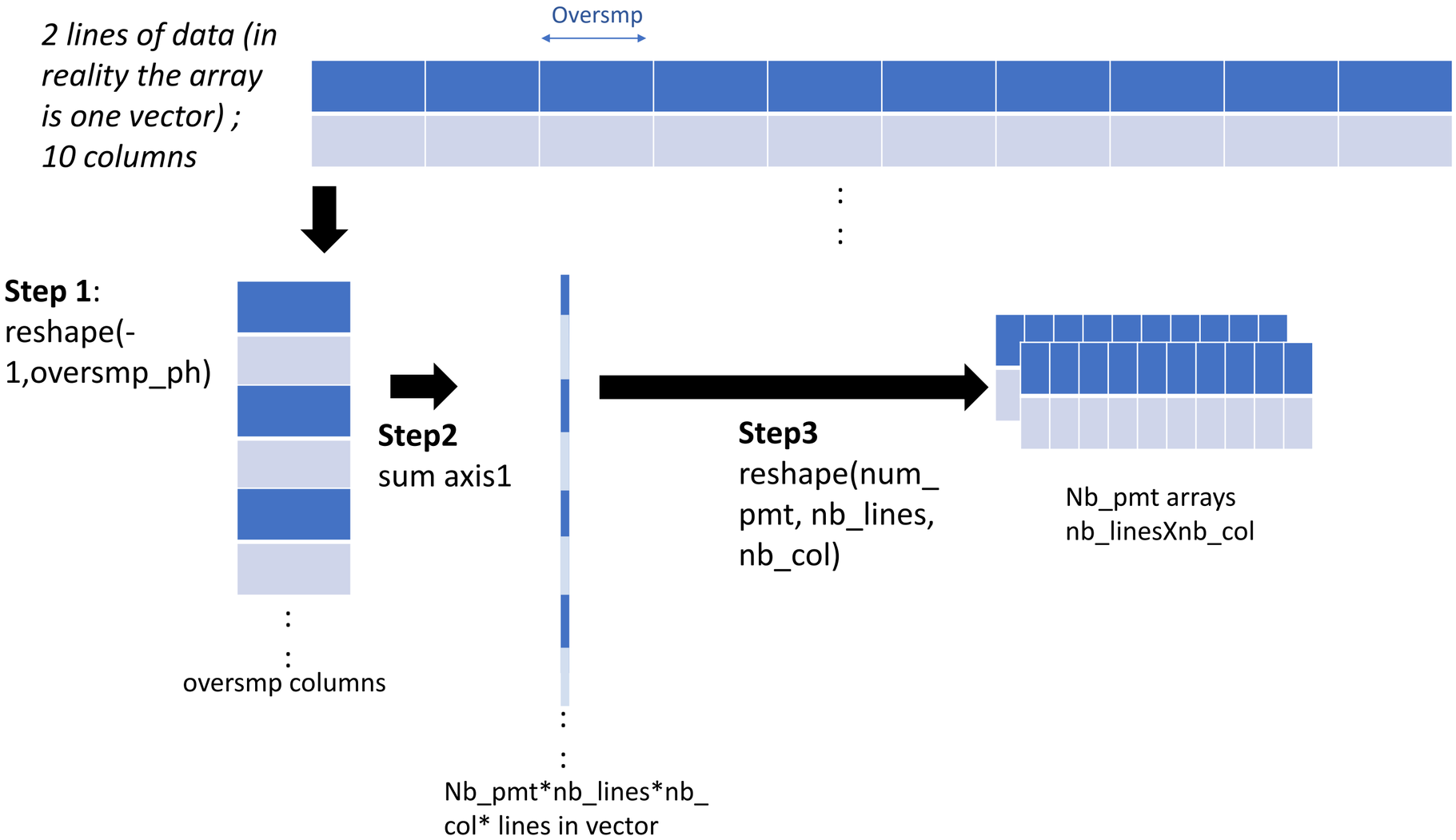
You could use get shift lines in Python, which uses Skimage. But it was showed to fail for large shifts.

Instead, in Matlab you can use DIPimage (install from website before). Using ‘reg_shift_advanced_func.m’ , the reg is done ~5 times to get accurate result.

[shiftv, im] = reg_shift_advanced_func(im0, lim1, off_shift, [])
%
% im0 image to treat (array of numbers), [] if load
% lim1 limit the ROI to X first lines (sizeY dflt)
% off_shift : final imposed offset (0 dflt)
% % !!! transpose the image to find shift on Y (here it's on X) !!!
% shift00 is [] dflt, unless to just shift

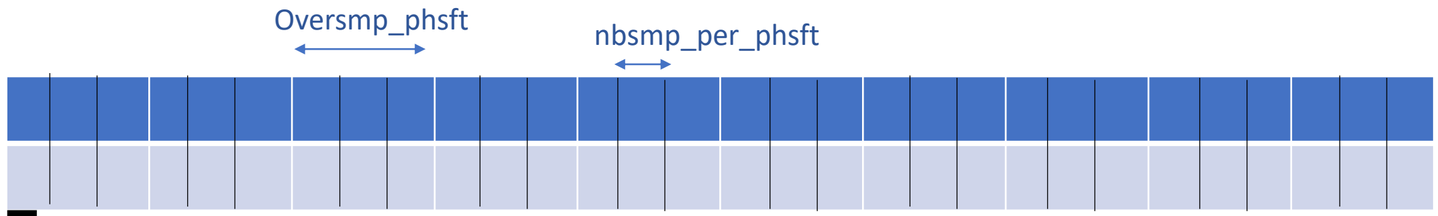
→ shiftv contains the shifts wanted. Works on X direction, for Y transpose image first !
→ It plots the result.

Fill array fast standard

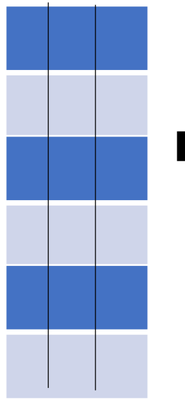


Fill array fast ishg

2 lines of data (in reality the array is one vector) ;
3 phsfts
10 columns

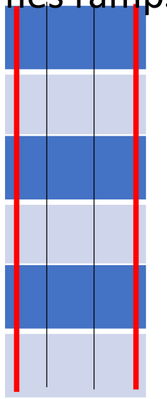


Step 1: reshape(-1,oversmp_ph)

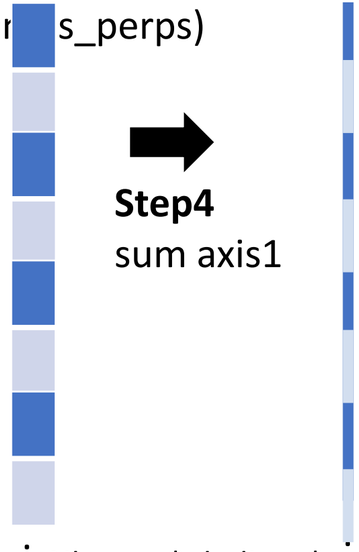


oversmp_ph columns

Step2: crop dead times ramps



Step3: reshape(nb_lines*nb_col*nb_ph*num_pmt, nb_sa s_perps)



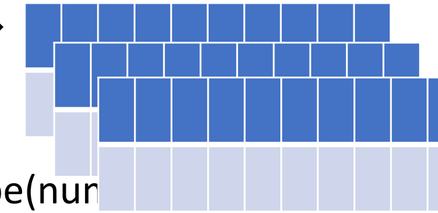
Nb_pmt*nb_lines*nb_col*nb_ph lines in vector

Step4 sum axis1

Step5 reshape(nb_col, -1)

Nb_pmt*nb_lines*nb_col lines and nb_ph columns

Step6 reshape(num_pmt, nb_lines, nb_col, nb_ph)



Nb_ph 3D arrays

Fill array fast standard/ishg (2)

TEST

```
num_pmt = 2; nb_lines = 4; oversmp_ph=6; nb_col=5; nb_ph = 3; nb_samps_perps = 2
numpy.set_printoptions(precision=1, threshold=10000, linewidth=150, suppress=True)
a=[1, 2, 3, 11, 22, 33, 111, 222, 333, 1111, 2222, 3333, 11111, 22222, 33333];
a1=numpy.array(a , dtype = numpy.float32) ; aa1=numpy.dstack((a1, a1+0.1, a1+0.2, a1+0.3)) .transpose((0,2,1)); aa1=numpy.squeeze(numpy.repea
aa1=numpy.stack((aa1, aa1),axis=0) # # creation
aa2 = aa1[:, :nb_lines*oversmp_ph*nb_col].reshape(-1, oversmp_ph);
aa3=aa2#aa3 = aa2[:, 5:-3]  # dead samples
aa4=aa3.reshape(nb_lines*nb_col*nb_ph*num_pmt, nb_samps_perps); # reshpr_samps_perps_ishg
aa5 = numpy.mean(aa4, axis = 1);
##aa6 = aa5.reshape( -1, nb_ph);
aa7 = aa5.reshape( num_pmt, nb_lines, nb_col, nb_ph); print(aa7[0, :,0, 0]) # reshper
```

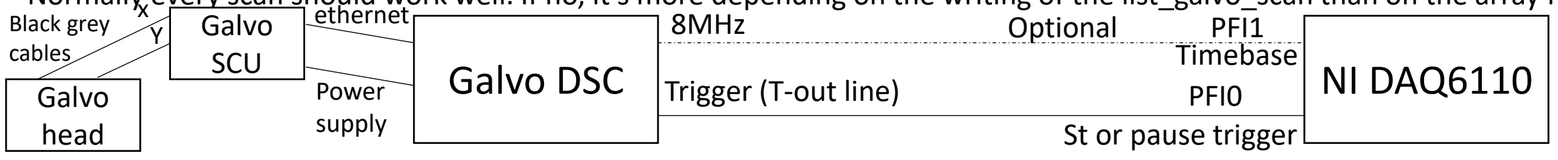
Config for dig galvos scan (1)

- 200x200_0.5um scan (400x400)

→ **With Pause trigger ctrl (line-time read):**

- Hardware: "Start trigger" of DSC unit connected to PFI0 (or read card) without 50Ω termination (with is less stable). Timebase 8MHz of DSC not used.
- Software: nb_skip (sync corr.) = 1.0. In bidirek, needs a 2.0 pixels offset in reverse. Not in unidirek. Any sample clock.

Normally every scan should work well. If no, it's more depending on the writing of the list_galvo_scan than on the array filling.



→ **Classic, start trigger and resync:**

- Hardware: "Start trigger" of DSC unit connected to PFI0 (or read card) (with or without 50Ω term).

Timebase 8MHz of DSC can be connected to PFI1 (of read card) , or not. If connected, "timebase_ext" must be checked on the GUI.

- Software: nb_skip (sync corr.) = 2.0.

If 8MHz ext. timebase is used, the sample clock must be a divider of 8MHz: 4MHz, 8/3MHz, 2MHz, 1MHz …

The effective sync corr. was found different with or without the master timebase, but this difference is corrected in the code by /10.

For every scan, the sync corr. must be found with this method. Some values are already pre-entered in the code of the gui_script.

Uses "fill_array_scan_digital2" in array_fill_loops.py

If a scan bugs, it is due to the writing of orders to the DSC. See next slide for a method to open the right VI in labview, define the scan and save it to .txt, to then load it in Python.

Use digital galvos by calling labview for defining the good scan (2)

- The digital galvos can be used in python, but have been tested only for 20us dwell time

The shift sync can be corrected for different ROI (see previous slide)
However, large dwell time like 200us are known to bug : don't use them !
→ Instead, use a (modified) VI of old soft labview that allows you to define the right scan : in folder "DefScanDigGalvo" on the Desktop, there are some shortcut to VIs: corr. folder is "MultiPHOTON – Copie"

a wrong COM is set for imic for the init to not take the control of any instrument : start the soft should raise an error « failed to open com port », just Continue

- open Main, and « edit/DefineScan » allows to set the scan parameters

The main challenge is the offsets, because you won't see them in labview !!
Change the scan, and click on "Change Scan"

- run a scan (or execute « DScanContinous » in Treiber folder), and it will write to the scan.txt the scan to send to python (you can verify the file, it's in Variables folder)
the Run is supposed to fail, because galvos are disconnected in this VI

- if run another time is not possible, you can just launch DScanContinous (RunControl) again
be sure that the variable scan.set is to 1. If errors just « Continue »

- When Exiting, always choose "don't save"

- If you want to avoid errors, try to modify the VI !

2019.07.23: implemented DLL for "findValidCycles.Vi" that is call by ctypes. Also, implemented "Galvo.vi" in DLL (calculate reversing time) for safety. Now the only differences remaining in the scan lists are because of center offset.

The DLL is in C:\Users\admin\Documents\Python\Packages\validcycles+flbk_x64: validcycles.dll (+lvanlys.dll)

Config for dig galvos scan (3)

```
galvo_rotation = 135 # deg
rotate = galvo_rotation*math.pi/180 # 0, rad
img_pixel = 500 # for labview, the size of the zone to define the scan
field_view = 0.006 # radius im Zwischenbild (intermediate image)
turn_time_unidirek = 0.0015 # # sec, as in labview
center_galvo_x00 = 2*-16.6442e-21 # corr. to zepto meter, center galvo SENSIBLE
center_galvo_y00 = 2*212.132e-6 # corr. to micro meter, is the double in LV ??
SM_cycle = 1e-5 # s
SM_delay = 119 # (number of cycles), galvo.delay, in Digital mode
scan_linse = 0.035 # in m, meaning 35mm focal for the SL
# really related to the galvos themselves
ohm_val = 4.1
induct_H = 9.8e-5
max_voltage = 23 # (V?)
inertia = 0.025 # g/cm^2
torque = 25000 # dyn cm/A
bit_size_galvo = 6.44e-7 #
bits_16_36 = round(2**36/2**16) #1048576
time_base_ext = 0 # define if the digital galvos use an external time_base or internal (change a bit the
sync value !!)
# if timebase_ext is 1
clock_galvo_digital = 8000000 # Hz # 10000000
min_timebase_div = 2 # in AD card
timebase_src_end = 'PFI1'
delay_trig = 0.000005 # 5e-6 s
trig_src_name_dig_galvos = 'PFI0'
max_offset_x_digGalvo = 2 # mm
max_offset_y_digGalvo = 2 # mm
dig_galvos_callback = 0 # specify if the galvo mode uses callback to read the samples each line
eff_loss_dig_galvos = 0.16 # in unidrek mode
```

```
# # ******************************************
# # ----- digital galvos connection parameters--
# # ******************************************


ressource_dig_galvos = 'COM8'
baud_rate_dig_galvos= 57600
# # bits_galvo = 8
# # parity_galvo = constants.Parity.none
# # stop_bit_galvo = constants.StopBits.one
# # flow_control_galvo= 0
timeout_dig_galvos = 2# sec 2000 ms
# # read_termination_dig_galvo = '\n'
# # write_termination_dig_galvo = read_termination_dig_galvo
```

- Optical: the beam has to come through the old galvos (bottom path), with the slider bottom not in position #2 (usually empty).

The beam can be aligned directly after the galvos_unit DSC + SCU has been open, and wait 1min for the end of initial vibrations (you can hear them).

Config for stage (sample) scan (1)

PORT COM (by USB): COM4
The USB must be set to work with a virtual COMport, see windows control panel (for change, reboot the unit).
Or work with FTDI instead of Pyserial.

- Hardware: Thorlabs XY BBD102 — CH1/2 USER IO — PFI2 (Start and ref trigger) — NI DAQ6110

- Optical: the beam is usually set to come with the silver mirror of slider_bottom (pos #2). The mirror has to be manually orientated for a beam parallel to X axis of stage. For this, open the slider and rotate the mirror. Other position is for anlg galvo scan.

Otherwise, the beam may come from the anlg or dig galvos that do not move, but it's unusual. The strength of stage scanning is that the beam do not cross a lot of optics.

- Software: it uses a start and ref trigger, with a digital trigger from the motor controller "max speed reached" (see ppt stage for details).

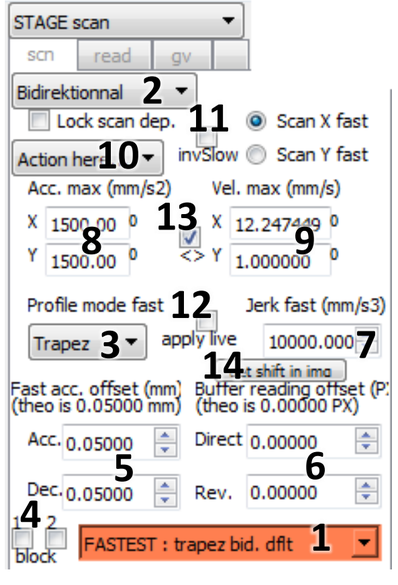
The GUI will normally set the optimal parameters, you should use these ones (exp. time not imposed). Different modes have different speed (**1**). The ones with "trapez" profiles are less reliable if the sample is not well fixed to the stage (glass coverslip or sample on it move a little bit).

→ **FASTEST:** they use trapez profiles. Bidirek is fastest, but needs a slight correction in pixel_offset (**6**). You can calculate the amount, or use (**14**) on a shifted image to get the amount by cross-correlation.

(**4**) Is for choosing if the motor waits at each move for completion, or not. To wait is safer but it's also longer. They are two blocks, 1 (block_each_step) is the main one (and 2 is on in this case): the code calls the blocking function each time. If 2 is on (force_wait_fast), it will not block directly but will read the "move_finished" message at some point. If no block, it reads the msg only at the end of the image (clearing buffer).

→**SAFEST:** they use S-curve profiles. Bidirek is fastest, but also may need a slight correction in pixel_offset (**6**). S-curve profiles do not allow non-blocking mode !!!
The jerk (**7**) is here important, it will be set to optimal value.

❑ You may change the exposure time, so the scan won't be the fastest one (for a given size) because the accn (**8**) or speed (**9**) won't be optimal. The good parameters will be updated though. You can always go back to the optimal parameters, or be sure they are well set by selecting "opt. params" in "action" (**10**). The connections of the many buttons may fail, so applying "opt. params" is a good habit.



Config for stage (sample) scan (2)

- (**1**) is an easy button for changing params, but every controls can be changed by other buttons. (**2**) is used for bidirek/unidirek mode. In (**5**) you may increase the acceleration offset to be sure that the stage has sufficient distance to do it (duration will increase).

(**11**): you can take the image from bottom to top, instead of beginning by top (inverse slow).

(**13**): couple or not accn and vel. of motor when setting them. If coupled, change on acceleration will change the speed. If not, possible to change the values independently.

(**12**): Change the motor parameters in live, and not only when a new scan begins. If you want to apply params now (vel., accn…), you can also choose “apply params!” in **10**.

(**10**): Apply params! : tells the stage to use current vel, accn etc.

Force update!: force the button to update to the new values (to be sure)

old scan: put params from previous scan.

Opt. params: set the parameters for fastest scan (changes exposure).

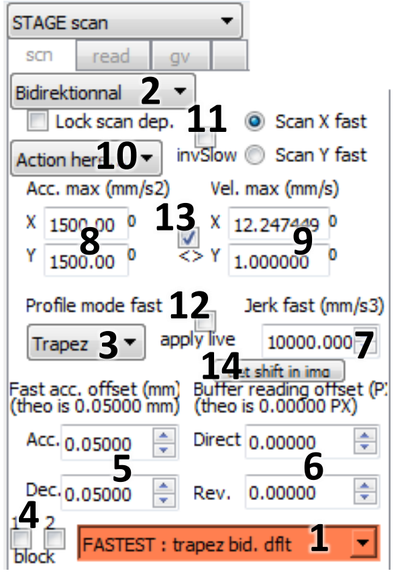
To move the motor XY in stage scan, you must wait for another scan (because the motor is in use). Otherwise, you can “kill scan Proc.” (button): the motor will then return to the entered position. After, it will be free so you can move it.
When exiting stage scan, the default acceleration and velocity will be (normally) be set back.

For moving the XY while scanning, it’s possible (but it may lead to a scan abortion due to bug).
If “Live X” is checked, the motor will move at next line of image. If not checked, it will move at next image.

→ The fastest mode with no block may lead to errors in the end, as all the “move complete” orders have not been cleaned. In that case, try to clean buffer. Try to reboot the stageXY Thread.
Try to reboot unit if it persists.

→If the scan you launched does not work, verify the connections. Look at the error message to identify the reason.
→Try to use a safest mode if possible. Try to decrease the speed, maybe.
→Try to reboot the motor controller, and Python.

→The scan parameters will be stored in the TIFF info of images:
Scan_params = ['unidirek', 'Xfast', 'buffPX_offsetdir/rev 0.0/0.0', 'acc_offsetdir/rev 1.00/1.00', 'vel_X/Y 21.5/4.0', 'accn_X/Y 464.2/1500.0', 'profile/jerk 2/10000', 'blocksteps 1&1']



Config for stage (sample) scan (3)

Sample clock is “onboard clock” of NI card, any sample rate should work (if scan is too big, it’s possible that the sample rate will be decreased). Dflt is 6110 card, NI6259 should also work, at sample rate <= 1MHz

Params in param_ini.py

use_serial_not_ftdi = 1 # 0 for USB ftdi
XYstage_comport = 'COM4'
trig_src_name_stgscan = 'PFI2'
prof_mode = 1 # dflt, 2 for S-curve and 1 for trapez
trigout_maxvelreached = 1 # 2 for 'In motion', 1 for 'Max Vel. Reached', 0 for off
prof_mode_slow = 1 # 2 for S-curve and 0 or 1 for trapez
jerk_mms3_slow = 10000 # mm/s3
jerk_mms3 = 10000 #10000001*2.2 #10000#0.0108 # mm/s3
jerk_mms3_trapez = 10000 # mm/s3, good default value
if jerk_mms3 > 10000001*2.2:
jerk_mms3 = 10000001*2.2
elif jerk_mms3 < 0.01:
jerk_mms3 = 0.01
time_out_stageXY = 0.0 # in s, waiting for completion uses 'while' loops, to listen for stop order or to wait for very long moves, so this value has to be small. it's not 0 because the reaction time of a human user is not infinitely short anyway, and a non zero value ensure a non too long while loop
block_slow_stgXY_before_return = 0 # for unidirek, block after slow movem, or not --> allow return fast move to start directly
min_posX = 0.01 # mm
max_posX = 109.99 # mm
min_posY = 0.01 # mm
max_posY = 74.99 # mm
bnd_posXY_l = [min_posX, max_posX, min_posY, max_posY]; max_val_pxl_l = [max_value_pixel_6110, max_value_pixel_6259]
acc_max = 1500 # mm/s2
acc_dflt = 500 # mm/s2
vel_dflt = 8 # mm/s
limitwait_move_time = 2 # sec, the limit of time user will consider waiting for an init move (in stage scan) without touching the motor speed/accn (if move time goes over it, the speed will be increased to dflt value and reset after move)

# # ------- PID parameters ---------
# For X
Kp_pos_val_toset = 300 # dflt 150
Ki_pos_val_toset = 175 # dflt 175
Ilim_pos_val_toset = 200000 # dflt 200000
Kd_pos_val_toset = 1000 # dflt 500
DerTime_pos_val_toset = 5 # dflt 5
OutGain_pos_val_toset = 6554 # dflt 6554
VelFeedFwd_pos_val_toset = 0 # dflt 0
AccFeedFwd_pos_val_toset = 1000 # dflt 1000
PosErrLim_pos_val_toset = 20000 # dflt 20000
# For Y
Kp_pos_val2_toset = 300 # dflt 65
Ki_pos_val2_toset = 175 # dflt 115
Ilim_pos_val2_toset = 200000 # dflt 200000
Kd_pos_val2_toset = 1000 # dflt 500
DerTime_pos_val2_toset = 5 # dflt 5
OutGain_pos_val2_toset = 3277 # dflt 3277
VelFeedFwd_pos_val2_toset = 0 # dflt 0
AccFeedFwd_pos_val2_toset = 1000 # dflt 1000
PosErrLim_pos_val2_toset = 20000 # dflt 20000
# default
Kp_pos_val_dflt = 150
Kp_pos_val_dflt_y = 65
Ki_pos_val_dflt = 175
Ki_pos_val_dflt_y = 115
Ilim_pos_val_dflt = 200000
Kd_pos_val_dflt = 500
DerTime_pos_val_dflt = 5
OutGain_pos_val_dflt = 6554
OutGain_pos_val_dflt_y = 3277
VelFeedFwd_pos_val_dflt = 0
AccFeedFwd_pos_val_dflt = 1000
PosErrLim_pos_val_dflt = 20000
tolerance_speed_accn_diff_real_value = 1.01 # # 1%

Other parameters in the code (acq_stage_script, …):

flag_reset_trig_out_each = False # for flyback
# # don't put to True as it will cause the code to bug if no blocking (method fast) !!
quiet_trig_initpos = False # # if you put True, the moving init will bug !
move_slow_before_flyback = False # does not really matter (for unidirek)
latency_fast = 25/1000 # in sec
nb_bytes_read_blck = 20
expand_data_to_fit_line = False # if less data than px, change the oversampling to stretch the data to the array
missing_samples_at_end_not_right = False # in stage scn, keep missing samples always on the right side.

Config for static scan

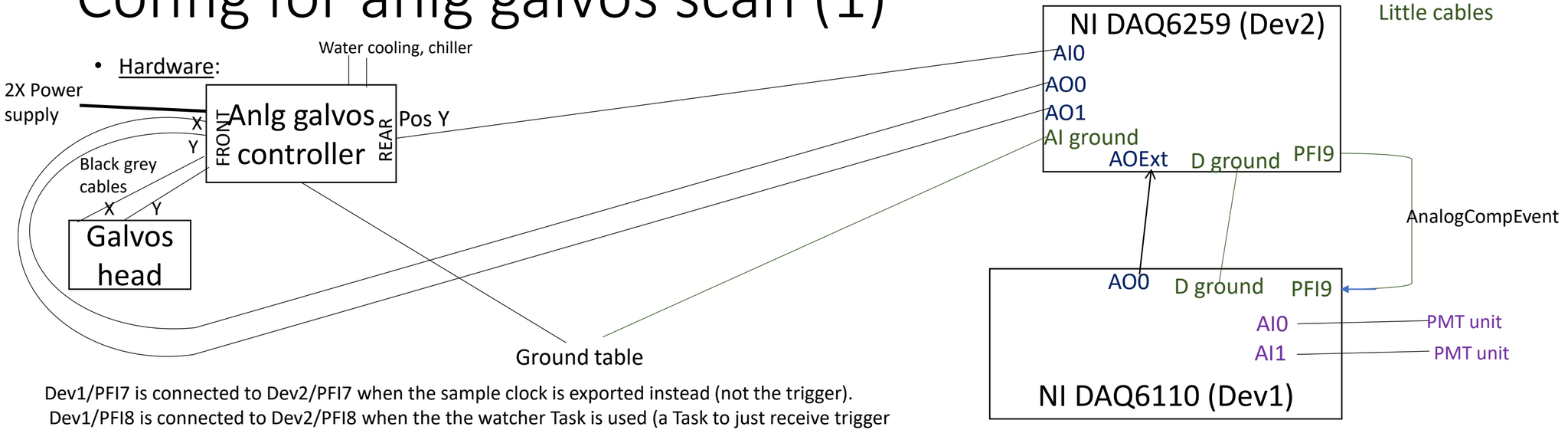
- Optical: the beam can come from the anlg, dig galvo, or stage path. It won't move any galvos or motor.
- Hardware: no trigger, so no connections. No instruments needed.
- Software: onboard clock, no trigger, any Nicard. Any sample rate.

Exemple of log window →

```
 fill Proc galvos started pr. wrkr
in worker_dataAcq. : NIDAQmx ok
In data_acquisition galvos, reading for
allocating buffer read ...
 ... buffer read allocated
buf_size AI read 5280528
actual_rate AI 4000000.0 src
/Dev1/ai/SampleClockTimeBase // timebase/
Dev1/Mastertimebase
AIChannel(name=Dev1/ai0)
 --- Summary ---
acq # 1
Buffer acq # 1/10
Buffer acq # 2/10
Buffer acq # 3/10
Buffer acq # 4/10
Buffer acq # 5/10
Buffer acq # 6/10
Buffer acq # 7/10
Buffer acq # 8/10
Buffer acq # 9/10
Buffer acq # 10/10
--- 3.23000431060791 seconds (acq.) ---
Expected time = 3.2 s
Nb loop done/expected in acq = 10/9.76
--- Fill time = 3.2 sec pr. ---
signal new img to disp sent to GUI via
In data_acquisition galvos, reading for
Order to stop detected in data_acquisti
In data_acquisition galvos, reading for
Order to stop detected in fill_process
Order to stop detected in disp_process
```

Config for anlg galvos scan (1)

Direct connections
Term 500Ω
Little cables



Dev1/PFI7 is connected to Dev2/PFI7 when the sample clock is exported instead (not the trigger).
Dev1/PFI8 is connected to Dev2/PFI8 when the the watcher Task is used (a Task to just receive trigger and send by Pipe).

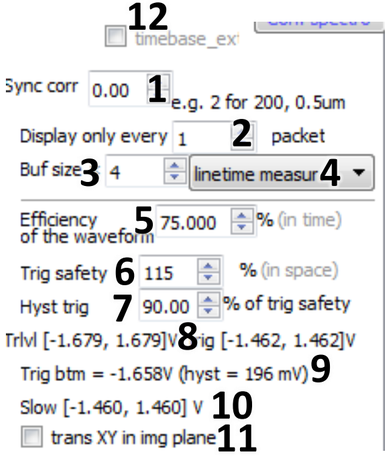
- Hardware:
- Software: onboard clock is used, the NI card might be inverted if needed. Only NIDAQ 6110 may be used, but in that case need to use directly PFI0 port as a trigger, which is less precise (8bits resolution, range cannot be set). Or use an AI channel as a trigger.

If Dev2 6259 is used alone, cannot use a trigger on AI channel, as the others AI will not be available. In that case, use APFI0 trigger, but less precise.

→ Method measure linetime: uses Dev2/ctr1 as the input counter for reading line times.

!! Some progressive offset observed with this method. For instance nb_skip = numpy.arange(0, nb_col, 0.288) for 22.8us/pixels or numpy.arange(0, nb_col, 2.05) for 250us/px ; 50x50um, 0.1um !!

→ Method callback each lines: uses Dev2/ctr0 to produce a counter output (that stays internal)

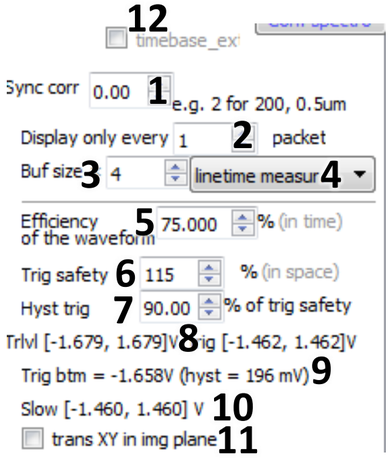


4 change between measure linetime/callback each lines. **1** nb_skip chamge (usually used for dig galvos. **2** buffer packets are displayed only every X ones.

3 buffer size has to be larger than one line time, this is the factor to multiply its size (for large exposure time, it was found more stable to put a high factor).

5 fraction of the total time spent in the direct ramp (1-eff spent in flyback). Unstable if > 80%. **6** factor to multiply the start and max positions, to be sure that the trigger threshold is passed (and not during a nonlinear part of the profile).

Config for anlg galvos scan (1b)



7 part of the amount of trigger safety used for hysteresis (if anlg LVL pause trigger and not window). The trigger will de-assert if it passes below threshold – hysteresis. To avoid many false triggering if there are oscillations.

8 Indications of volt positions that are written to galvos ; trigger lvls that must be input to terminal given chosen lvl in DAQ Task. **9** (trigger - hyst) to input to port, and hyst. level in real V. Red indications means that the volt parameters are surely outside the safe zone for galvos, will maybe lead to some bug. Reduce the offset X, or change the parameters. **10** indications of volt written to slow pos. **12** not often used, if use an external timebase or not (to go with lock-in for instance).

❑ Changing the exposure time may lead to a change of **3** , be sure that the value is correct. Also, if **5** is too slow the scan will be too long.

unirek_skip_half_of_lines = False if anlg LVL pause trigger and not window.
Anlg galvos uses the fast mode to fill arrays, with get_dur_lines if measure line time but not if callbacks.

→ Method callback each lines was found more stable for line fill in arrays, because the linetime measured varies a lot. Maybe works less well if the linetime is very small, because the callback function can be called at maximum kHz rates !
→ Line time measures also works if the synchro is not a problem.

You might want to translate the position in the XY plane of galvos, if so check box **11** : it will translate both X and Y pos to match the galvos' angle

Config for anlg galvos scan (2)

Param_ini.py

num_dev_AO = 1 # the device to write samples to move the galvos
num_dev_anlgTrig = 1 # the device used to just convert the analog trigger into a digital trigger
num_dev_watcherTrig = 1 # the device used to callback if trigger paused, or control trigger temporal width
term_trig_name_digital = 'PFI0' # can be any PFI
term_trig_name_6110 = term_trig_name_digital # on 6110, the PFI0 port can be used for analog AND digital triggers
term_trig_name_6259 = 'APFI0' # on 6259, the APFI0 port can be used for analog trigger (digital is on PFI0)
term_6110_clckExt = 'PFI7'
term_6110_trigExt = 'PFI9'
term_6259_clckExt = 'PFI7'
term_6259_trigExt = 'PFI9'
term_DI = 'port0/line0' # for 6259 only available, 6110 does not support changedetection, but it could be done with its sample_clock (on any PFI)
term_do = 'port0/line0' # # for 6259, available are any PFI, CTR1 OUT, CTR0OUT, FREQ OUT, P0. For 6110, any P0.0-7
trig_src_end_term_toExp_toWatcher = 'PFI8' # used only if different terminals have to be used for export to watcher and AI (usually unused)
trig_src_end_toExp_toDIWatcher = 'PFI5' # # for digital input (not very used)
method_watch = 4 # 4 for counter out, 1 for chg detect (6259 only), 6 for counter input with possibility of dig. filter
# 7: # counter input to MEASURE the line time; # # !! is already changeable on the GUI's front-end
# 6: # counter input for callback, that counts the falling triggers edges
# 4 counter OUTPUT retriggerable that makes a pulse (for callback) each time the st trigger = pause trigger of read task is asserted
# 5 -
# 3 anlg trig watches itself (2 cards): callback on sample clock of an AI task (on other card), whose clock is the analogComparisonEvent of the main read
# 2 : DI with sample clock detect (callback on sample clock), has the drawback to have to set the rate, FOR 6110 only
# 1 : DI with a callback on CHANGE_DETECTION_EVENT # for 6259 only
use_callbacks = 1
export_smpclk = 0
export_trigger = 1
DI_parallel_watcher = 0 # use a digital input that watch the trigger and callback a send via Pipe when change on it
DO_parallel_trigger = 0 # use a home-made pause trigger : an alg Task monitor it, and send a DO that acts as a digital trigger
use_chan_trigger = 1 # 0 to use a terminal port, 1 to use a channel
lvl_trigger_not_win = 2 # 1 for lvl anlg, 0 for window anlg, 2 for reject only vibration on top of the waveform
safety_lvl_trig_max_fact = 1.05 # fact to divide the max anlg lvl thres.
safety_fact_chan_trig = 1.1 # for the bound of anlg trigger read, if channel
# # triggerAddSafeFactor_imposed = 2# 1.1 # multiply by this factor the target voltages to be sure to get eventually outside the scan window
# # smp_rate_trig = 1e6 # sample rate used if an independant AI Task is anlg triggered, and export its trigger
use_dig_fltr_onAlgCmpEv = False #False #False # use a digital filter on the counter output that watch the trig for it not to callback when jitter
use_trigger_anlgcompEv_onFallingEdges = True # using falling edges avoid inversion of terminal
use_diff_terms_expAI_expWatch = False #False ## use explictly different terminals to export trigger signal to AI and to watcher (from trigCtrl). Use it if it improves
add_nb_lines_safe = 2 # # ask the soft to read the time of N lines, but acquire samples of n+2 lines to be sure enough samples are acquired
hyst_trig_min_advised = 20/1000 # V, min. hysteresis for lvl trigger
if num_dev_anlgTrig == 0: # 6110
if use_chan_trigger:
factor_trigger = factor_trigger_chan
else: # use a terminal port PFI0 directly
factor_trigger = 1/0.35
else: # 6259
if use_chan_trigger:
factor_trigger = 1.97
else: # use a terminal port APFI0 directly
factor_trigger = 2.09 #2 #2.09 #2.55
angle_rot_degree_new_galvos = 0 #45
#!! If posX is used as trigger, need to re-define the trigger level(s)
#every line, which cannot be done while the Task is running !
use_velocity_trigger = 0 # not use pos but vel. trigger
force_buffer_small = 1 # buffer the smallest possible (according to fact) : can avoid latencies
fact_buffer_anlgGalvo = 4 # buffer small is fact*nb_samps_line : has to be at least 2
time_buffer_tobeCorrect = 1 # sec, if large buffer chosen : buffer AI can contain samples max up to this duration (rate dependant)
sample_rate_min_6110 = 0.1e6 # 100kHz is minimum rate for 6110
sample_rate_min_6259 = 0 # 0 is minimum rate for 6259

Config for anlg galvos scan (3)

Param_ini.py (suite)

```
# # --------- AO write params ------------------
min_val_volt_galvos = -10 # V
max_val_volt_galvos = 10 # V
# The system is set for +-10V <-> +-10° mechanical
safety_ao_gen_fact = 1.05 # mult. the max expted range by this val 2be safe (for 6259)
ext_ref_AO_range = True # for 6259, use an external src on APFI0 for determining the range of AO
generation: need another AO (from 6110 ?) to supply a voltage
if (not use_chan_trigger and ext_ref_AO_range): # cannot be used if APFI0 is used as anlg trigger
ext_ref_AO_range = not ext_ref_AO_range
offset_y_deg_00 = 0 #0.8 #0.25 #0.75 # fast
offset_x_deg_00 = 0#-0.3 # -0.3 #0.35 # slow
use_volt_not_raw_write = 0
repeatability_galvos_V = 8e-6/math.pi*180 # in ° or V
timeout_galvo = 2000 # ms
bits_write = 16 # always, for both cards
nb_ao_channel = 2 # X and Y
tolerance_nb_write_to_stop = 2 # it's possible that the AO Task won't output all the samples
requested, so set a tolerance to consider the Task as done (2 is good)
duration_scan_prewrite_in_buffer=0.5 # sec, if write in LIVE
security_noise_factor = 1.15 # empirical
small_angle_step_response_us = 200 # us
smAngStpResp_safeFac = 5 # fact for small angle step response in reality
settling_time_galvo_us = smAngStpResp_safeFac*small_angle_step_response_us
# # us, Ts is the 0 – 99% settling time for a critically damped system to a step input.
divider_settling_time_avoid_step = 25 # must be between 20 and 100
lower_bound_divider_settling_time_avoid_step = 20
upper_bound_divider_settling_time_avoid_step = 100
limit_small_step_angle_measured = 0.8 # ° or V
smpRate_SafFact = 1 # empirical
sample_rate_AO_min_imposed = 1/small_angle_step_response_us*1e6*smpRate_SafFact #
always the same rate, goes faster does not help long exposure times, for short exposure times the
limit is just the angle range and the small_angle_step_response
BW_full_scale_galvos = 200 # Hz, for a square wave, or sawtooth, or triangular
BW_small_steps_galvos = 1000 # Hz
small_angle_step = (max_val_volt_galvos - min_val_volt_galvos)/100 # 1%

# # ***********************************************
# # ------- galvos hardware --------------------
# # ***********************************************
rotor_inertia = 0.125 # gm*cm2, +/-10%
mirror_inertia = 0.3 # gm*cm2
total_inertia = rotor_inertia + mirror_inertia
torque_constant = 6.17e4 # Dyne-cm/Amp, +/-10%
induct_coil_H = 180e-6 # 180 µH, +/-10%
ohm_coil_val = 2.79 # Ohms, +/-10%
max_rms_current_one_axis = 3.9 # Amp, see specs of 6220H
max_ddp = max_val_volt_galvos # ohm_coil_val*peak_current #
revers_time_meth = 2 # 2 = imposed ; # 0 like the digital galvos, and 1
calculated with theory
# calculate max acceleration of galvos using the peak current (1), or
calculate the optimal (0), or impose it (2)
scan_lens_mm = 45 # the scan lens just after the galvos to demagnify
the scan and expand the beam (with tube lens)
ai_readposX_anlggalvo = 'ai2'
ai_readposY_anlggalvo = 'ai3'


# # ***********************************************
# # --------- scan params ------------------
# # ***********************************************
smallest_OS_sleep = 20e-3 # on windows smallest sleep can be 13ms
nb_lines_acumm_acq = None # None means that the code will take the value
calculated with update rate
# # NOt used (temporarily ??)
shape_reverse_movement = 0 # add some points on the reverse of galvos speed,
to smooth
correct_unidirektionnal = 1 # 1 for a smooth return, 2 for acting as if it was a
bidirek scan
skip_first_read = 0 # when the acq. bugs, it's often due to 1st line so this
command just throw it away
blink_after_scan = 1
volt_pos_blink = -4.9 # min_val_volt_galvos # voltage to set at each new img
beginning
fact_data = 2 # size of the array to contains line = fact*nb_samples_expected
```

Config for anlg galvos scan (4)

LOG WINDOW Scan with meas. linetime

in worker_dataAcq. : NIDAQmx ok
In data_acquisition galvos, reading for job if any ...
be sure that AnlgCompEvent of the Device(name=Dev2) that trigs is connected to t
he Device(name=Dev1) that reads AI, and that its trigger is lvl digital
SignalModifiers.INVERT_POLARITY
I connected /Dev2/AnalogComparisonEvent to /Dev2/PFI9
allocating buffer read ... ... buffer read allocated
buf_size AI read 5280528
actual_rate AI 4000000.0 src /Dev1/ai/SampleClockTimebase // timebase /Dev1/Mast
erTimebase
AIChannel(name=Dev1/ai0)
Warning: Dumb Task on Dev1 is giving AO range to Dev2/ao0:1 (verif cable): 2.0!! -2.0 2.0
write_scan_before 1
buf_size AO 122000
actual_rate AO 25094.10288582183
I have been to init pos
Space remained in buffer 0 / 122000
--- 0.040000 seconds write time
 --- Summary ---
device to AO Dev2, device to read Dev1, dev to trig Dev2, dev to watchtrig Dev2
 export trigger 1, export smp clk 0
 use_callbacks 0, method_watch 7,
read input START trigger : TriggerType.NONE
read input PAUSE trigger DIGITAL on src /Dev1/PFI9 (slave) with pause when Level.LOW
Trig input trigger Anlg LEVEL on src (alone) Dev2/ai0 with pause (lvl LOW) when
ActiveLevel.ABOVE hysteresis 0.099609375
TrigWatcher pulse width Dev2/ctr0 measure on /Dev2/AnalogComparisonEvent and dig
 fltr is False and time 0.000 msec
acq # 1
Buffer acq # 1/15
...
Buffer acq # 15/15
stopped AO
--- 4.89000678062439 seconds (acq.) ---

LOG WINDOW Scan with callbacks each line

be sure that AnlgCompEvent of the Device(name=Dev2) that trigs is connected to t
he Device(name=Dev1) that reads AI, and that its trigger is lvl digital
SignalModifiers.INVERT_POLARITY
I connected /Dev2/AnalogComparisonEvent to /Dev2/PFI9
allocating buffer read ...
 ... buffer read allocated
buf_size AI read 5280528
actual_rate AI 4000000.0 src /Dev1/ai/SampleClockTimebase // timebase /Dev1/Mast
erTimebase
AIChannel(name=Dev1/ai0)
Warning: Dumb Task on Dev1 is giving AO range to Dev2/ao0:1 (verif cable): 2.0!! -2.0 2.0
write_scan_before 1
buf_size AO 122000
actual_rate AO 25094.10288582183
I have been to init pos
Space remained in buffer 0 / 122000
--- 0.040000 seconds write time
 --- Summary ---
device to AO Dev2, device to read Dev1, dev to trig Dev2, dev to watchtrig Dev2
 export trigger 1, export smp clk 0
 use_callbacks 1, method_watch 4,
read input START trigger : TriggerType.NONE
read input PAUSE trigger DIGITAL on src /Dev1/PFI9 (slave) with pause when Level.LOW
Trig input trigger Anlg LEVEL on src (alone) Dev2/ai0 with pause (lvl LOW) when
ActiveLevel.ABOVE hysteresis 0.099609375
TrigWatcher input START trigger DIGITAL on src (slave) /Dev2/AnalogComparisonEvent
acq # 2
sent buffer # 1 (27 lines)
...
sent buffer # 14 (27 lines)
waiting for AO Task to finish generate ...
... AO is over
sent buffer # 15 (22 lines)
Task: watcher_01
--- Fill time = 4.9 sec pr. ---
signal new img to disp sent to GUI via go_galvo
diff nb_el ./ mean (%) [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Config for anlg galvos scan classic SOFTWARE (5)

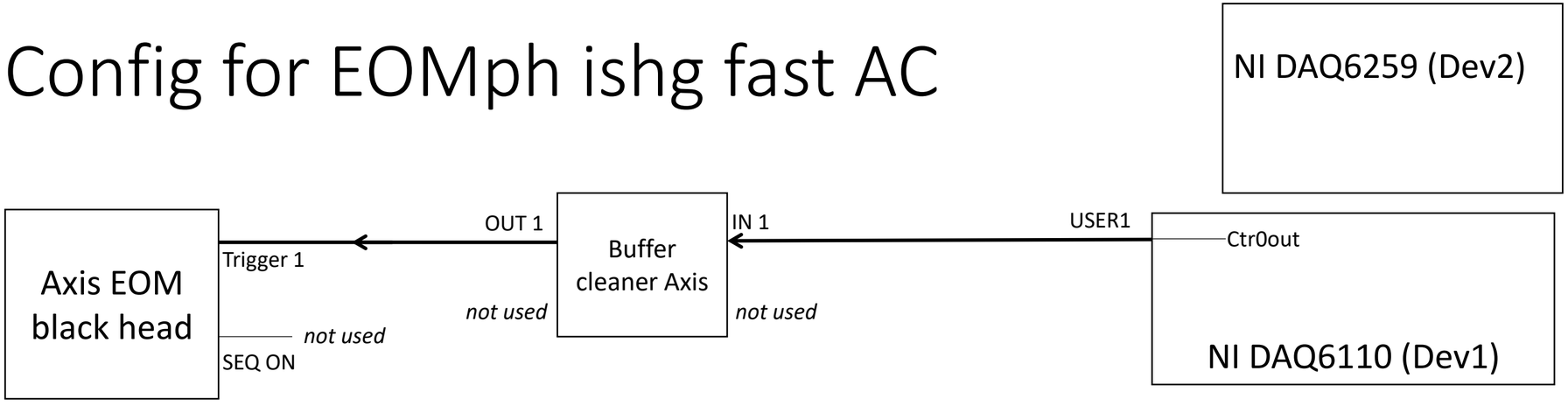
- Uses Dev2 to receive the analog trig (with an AI dumb Task), and produces a digital trig : device_to_use_anlgTrig

- Uses Dev2 to ‘watch’ the trig, also with a Task (method-dependent: counter I/O, …): device_to_use_watcherTrig.

- Uses Dev2 for AO to write to galvos the scan: device_to_use_AO.

- Uses Dev1 to read AI normally: device_to_use_AI. The AI receives a digital trigger from the dumb AI Task of Dev2.
  The lines separation is ensured by the watcherTrig, that receives the anlgcompEvent of the dumb AI Task. It either measures the line times, or register callback with some events (trigger edge falling …).

Config for EOMph ishg fast AC



Stage : Dev1/PFI2out trigger to internal pass Dev1/PFI4, that serves as a trigger for a CO rettrigerable Task on Ctr0Out

Galvos : /Dev1/PFI9 trigger to internal pass Dev1/PFI4, that serves as a trigger for a CO rettrigerable Task on Ctr0Out

Both cases use a CO task to produce a pulse of 0.4us up time and 0.4us low time
But for galvos callback, a CO Task is already used to watchtrig (and callback) ! So dig galvo uses the same CO,
and do not need ctr_mtrout Task or PFI4 pass.

For some reasons it does not work with anlg galvos (which uses Dev2/Ctr0out for callback), so the ctr_mtrout Task is used.

Config for EOMph ishg fast ANLG GALVOS measlinetime (log)

```
 ... buffer read allocated
buf_size AI read 6600660
actual_rate AI 5000000.0 src /Dev1/ai/SampleClockTimebase // timebase /Dev1/MasterTimebase AIChannel(name=Dev1/ai0)
nb phshft 32 , divided by 2:True; 3:False; 4:True; 5:False;
 warning, imposed step phsft leads to a nb_phsft that was not adujstable within
tolerance of 20 samps: first 4 samps of ramp (meaning 40.0°) will be croped !
!! croped from ramp 0+4 samps !!
 3 samps per phasepxl exp_ph 0.60 us step 29.08° optimized, closest
counter OUT ishg Dev1/ctr0 with trig /Dev1/PFI4 pulsewidth_meas:0.4 (4e-07, 4e-07) us
ishg : connected /Dev1/PFI9 to /Dev1/PFI4 ['mtr_trigout_02']
 PFI4 is just an internal pass, connect Dev1/ctr0 OUT to buffer !
Warning: Dumb Task on Dev1 is giving AO range to Dev2/ao0:1 (verif cable): 2.0!!  -2.0 2.0
write_scan_before 1
buf_size AO  139200
actual_rate AO 25157.232704402515
I have been to init pos
Space remained in buffer  0 / 139200
--- 0.030000 seconds write time
ishg_EOM_AC_insamps [11, 100, 32, 260, 1400, 3, (4, 10, 4), (False, 114)]
 flag, nb_samps_ramp00, nb phsft, Vpi, VMax, nb_samps_perphsft, offset_samps, fl
ag_impose_ramptime_as_exptime sec+prim_proc False
 --- Summary ---
device to AO Dev2, device to read Dev1, dev to trig Dev2, dev to watchtrig Dev2
 export trigger 1, export smp clk 0
 use_callbacks 0, method_watch 7,
read input START trigger : TriggerType.NONE
read input PAUSE trigger DIGITAL on src /Dev1/PFI9 (slave) with pause when Level.LOW
Trig input trigger Anlg LEVEL on src (alone) Dev2/ai0 with pause (lvl LOW) when
ActiveLevel.ABOVE hysteresis 0.099609375
TrigWatcher pulse width Dev2/ctr0 measure on /Dev2/AnalogComparisonEvent and dig
 fltr is False and time 0.000 msec
acq # 2
Buffer acq # 1/17
Buffer acq # 2/17
estimated fill ishg 2.7 sec
…
```