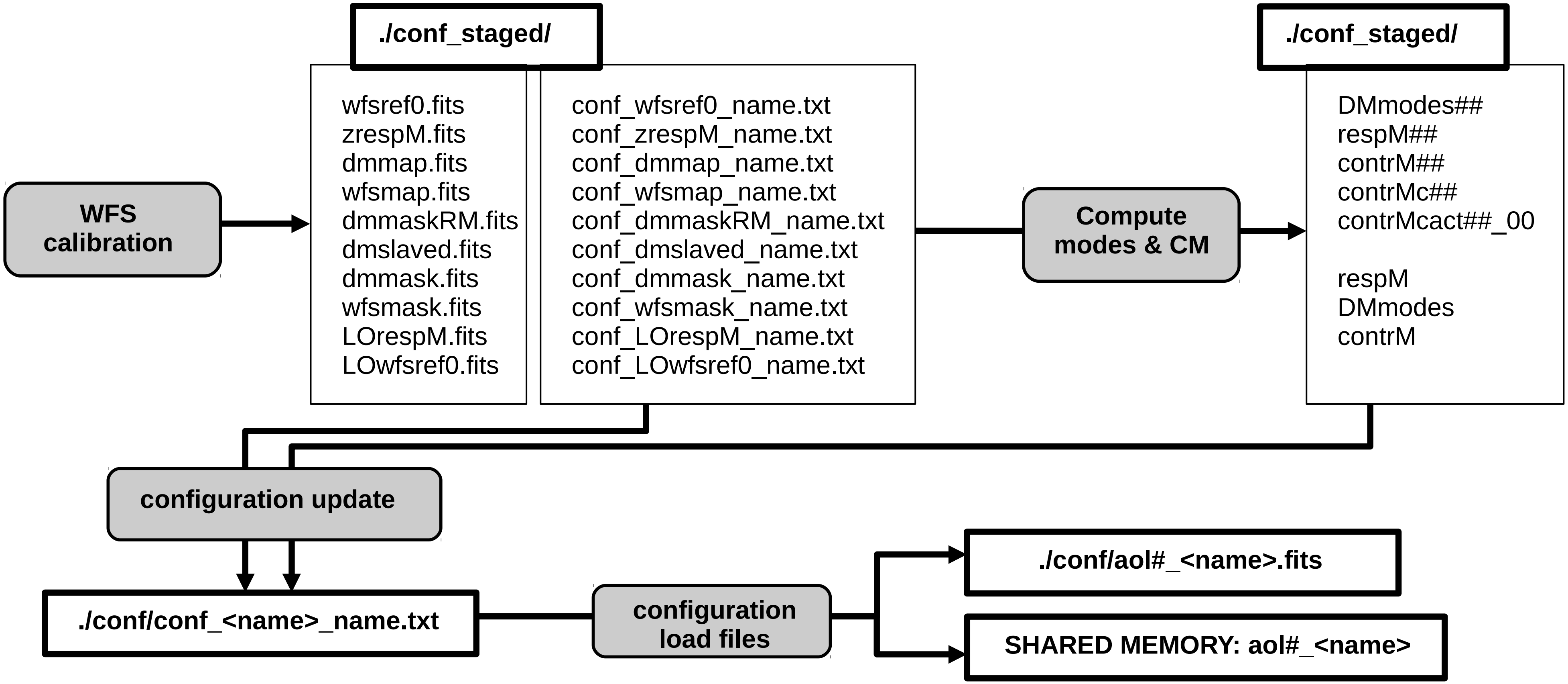
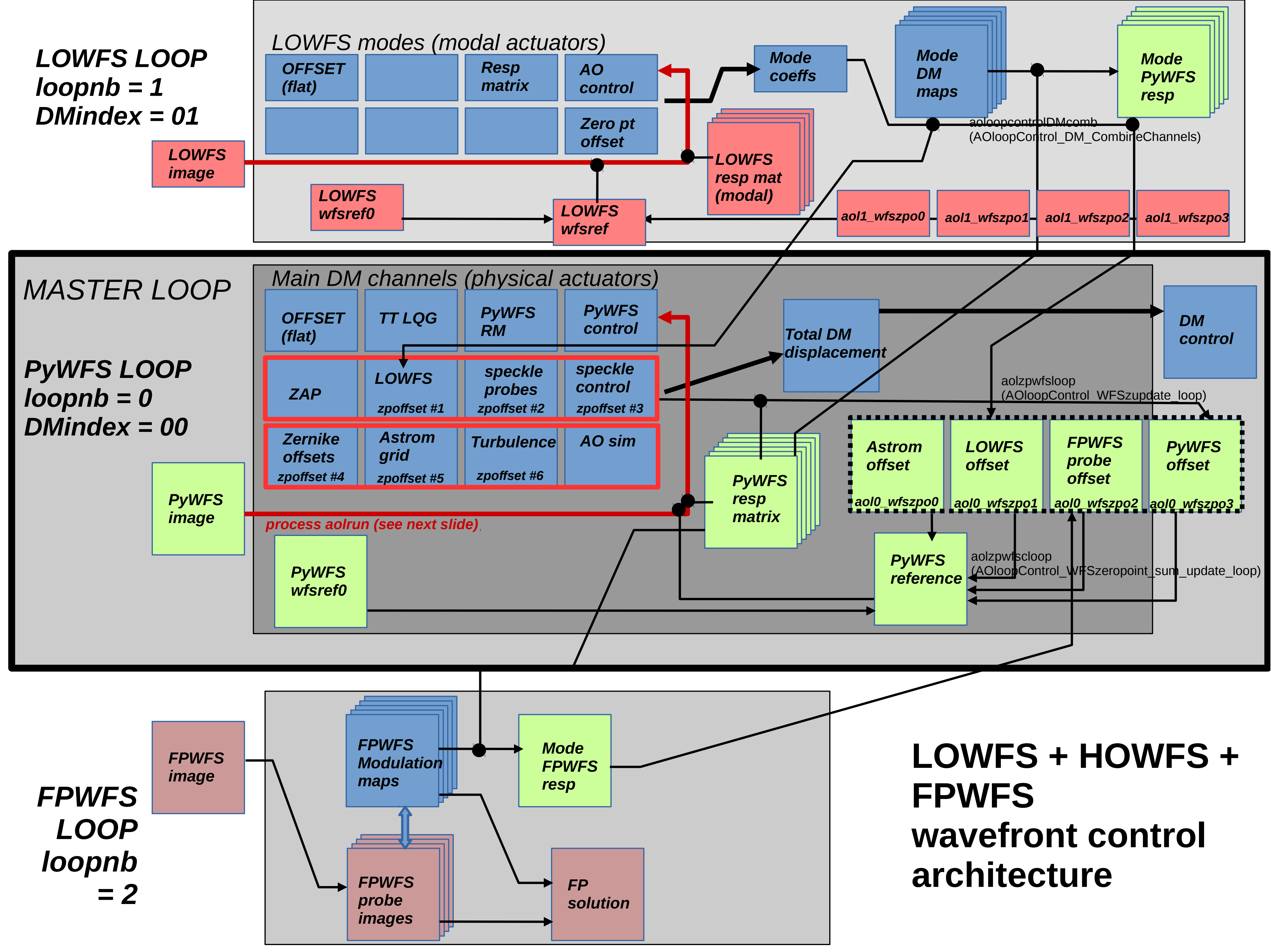
For each file:
conf_<name>_name.txt points to archived file location

conf/conf_<name>_name.txt are read by function ReadConfFile for loading into shared memory and FITS copy to ./conf/aol#_<name>.fits

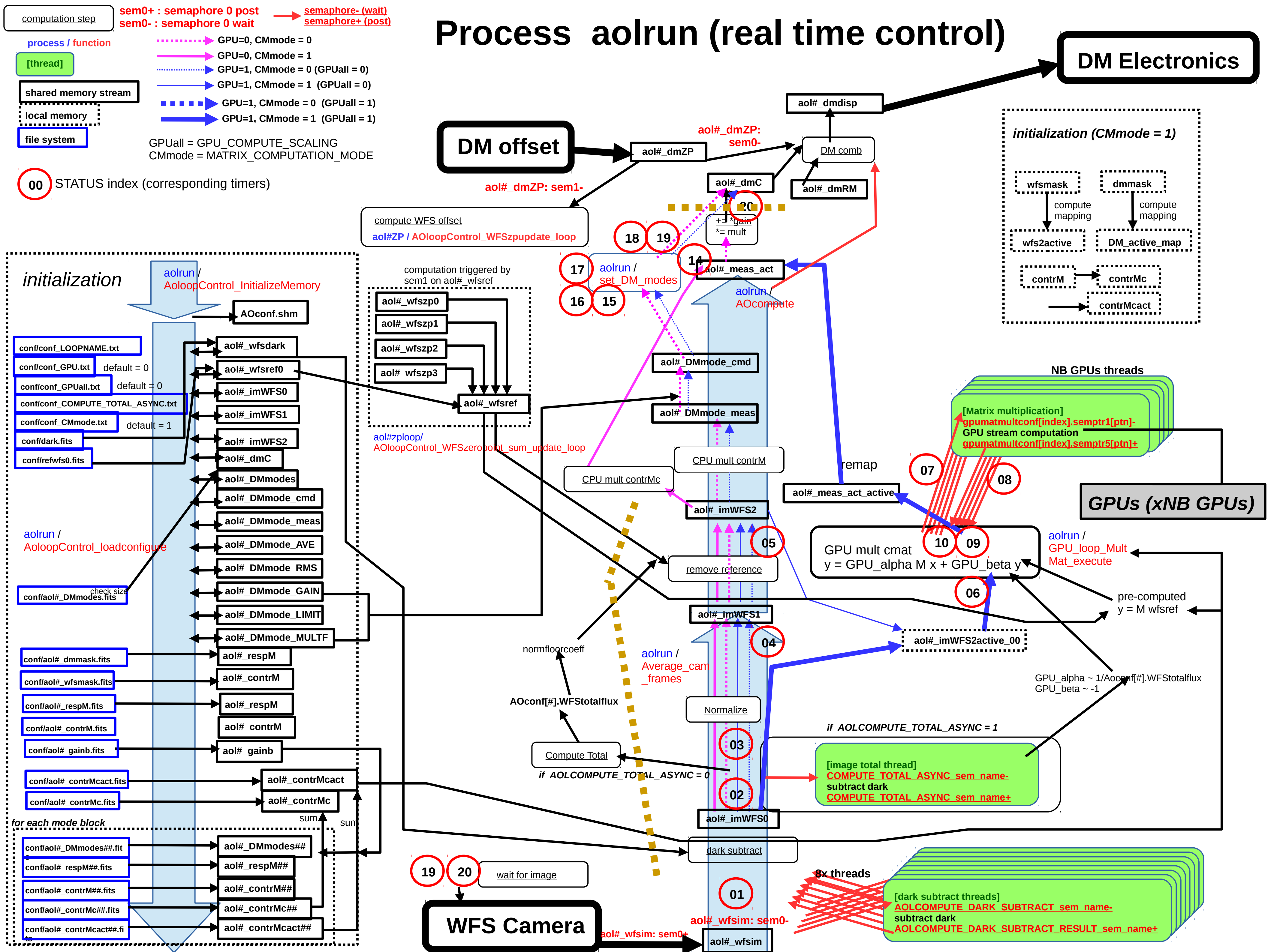
Work Flow





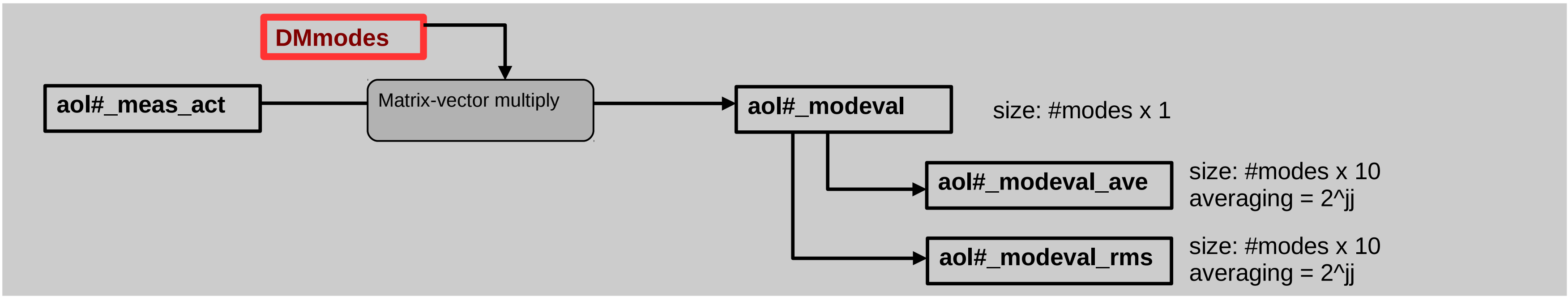
LOWFS LOOP
loopnb = 1
DMindex = 01
LOWFS modes (modal actuators)
OFFSET (flat)
Resp matrix
AO control
Zero pt offset
Mode coeffs
Mode DM maps
Mode PyWFS resp
aoloopcontrolDMcomb
(AOloopControl_DM_CombineChannels)
LOWFS image
LOWFS resp mat (modal)
LOWFS wfsref0
LOWFS wfsref
aol1_wfszpo0
aol1_wfszpo1
aol1_wfszpo2
aol1_wfszpo3
MASTER LOOP
Main DM channels (physical actuators)
OFFSET (flat)
TT LQG
PyWFS RM
PyWFS control
ZAP
LOWFS
zpoffset #1
speckle probes
zpoffset #2
speckle control
zpoffset #3
Zernike offsets
zpoffset #4
Astrom grid
zpoffset #5
Turbulence
zpoffset #6
AO sim
Total DM displacement
DM control
PyWFS LOOP
loopnb = 0
DMindex = 00
aolzpwfsloop
(AOloopControl_WFSzupdate_loop)
Astrom offset
aol0_wfszpo0
LOWFS offset
aol0_wfszpo1
FPWFS probe offset
aol0_wfszpo2
PyWFS offset
aol0_wfszpo3
PyWFS image
process aolrun (see next slide)
PyWFS resp matrix
PyWFS wfsref0
PyWFS reference
aolzpwfscloop
(AOloopControl_WFSzeropoint_sum_update_loop)
FPWFS image
FPWFS Modulation maps
Mode FPWFS resp
FPWFS probe images
FP solution
FPWFS LOOP
loopnb = 2
LOWFS + HOWFS + FPWFS
wavefront control
architecture

Process aolrun (real time control)

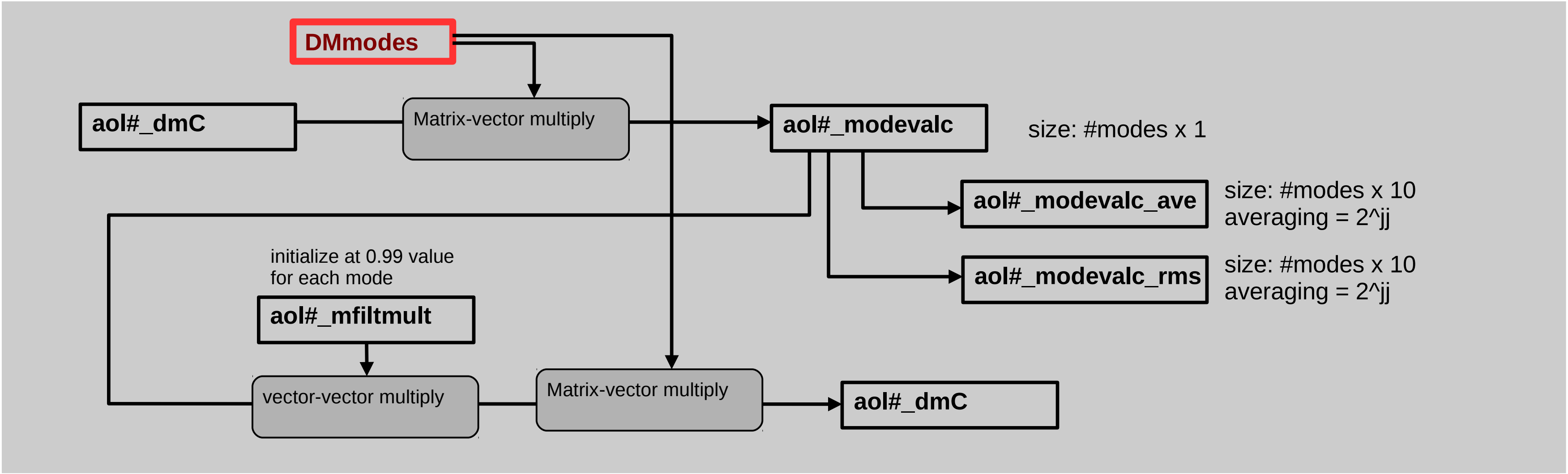


Auxillary processes

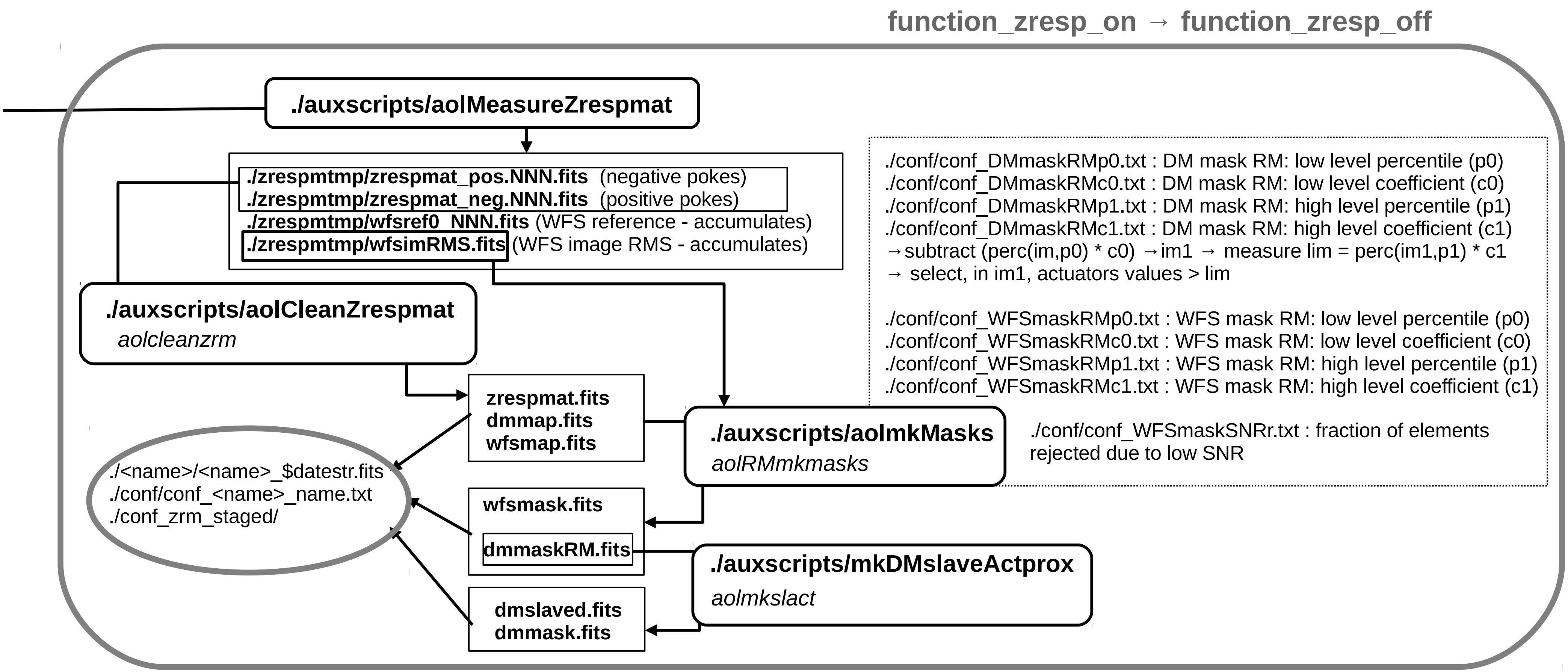
Decompose WFS measurements in modes

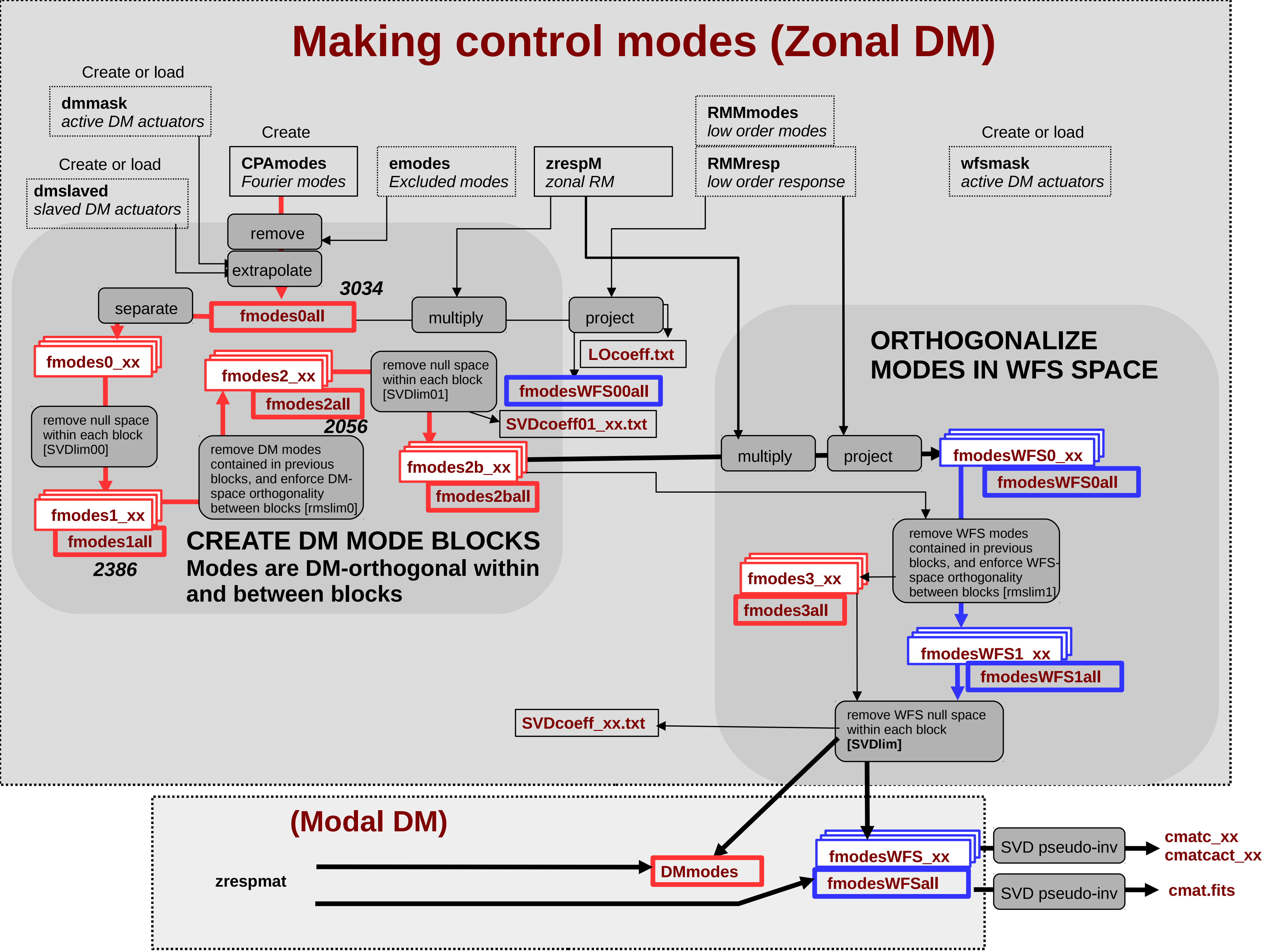


Decompose DM commands in modes + apply modal mult gains



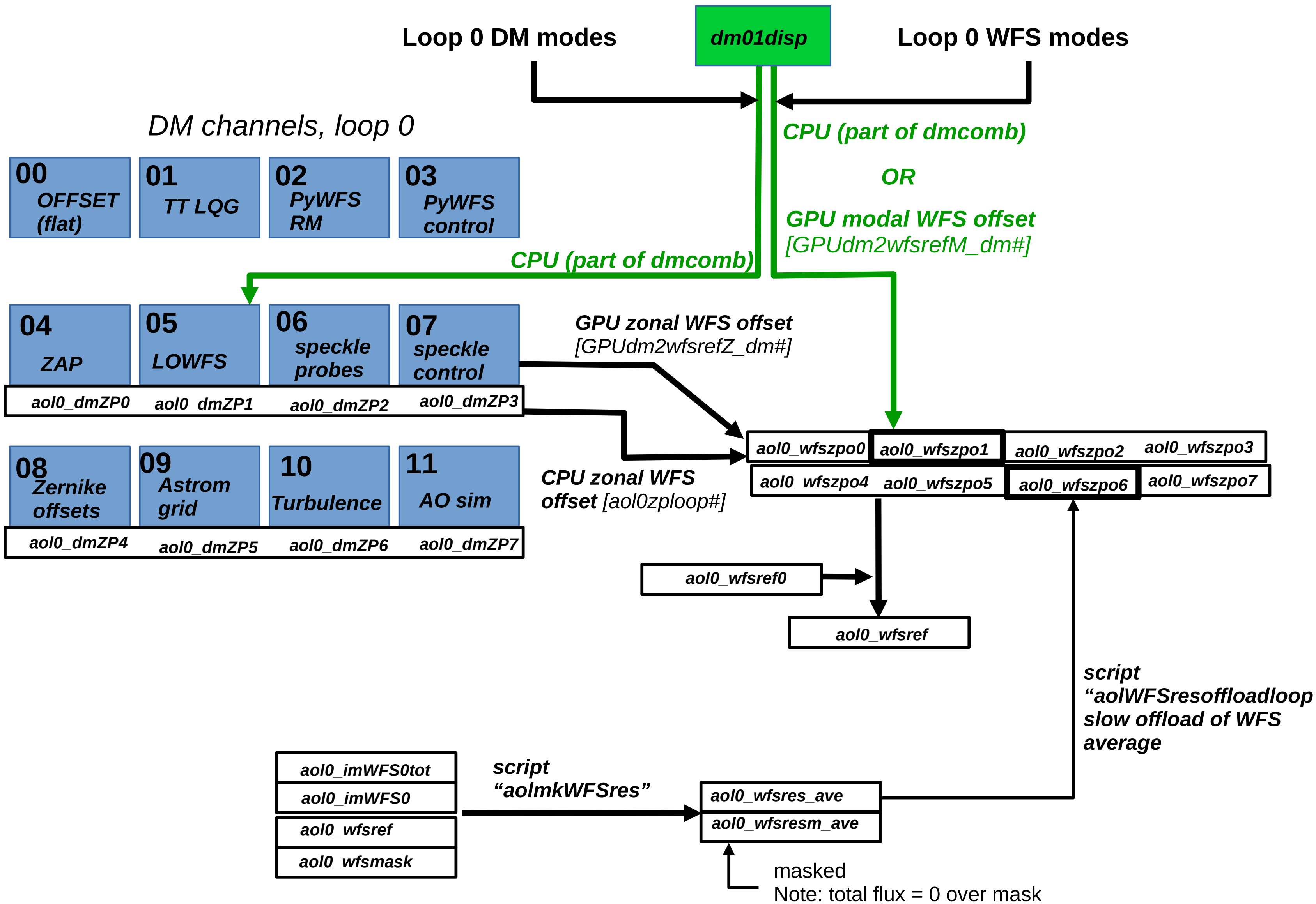
Zonal response matrix acquisition → masks



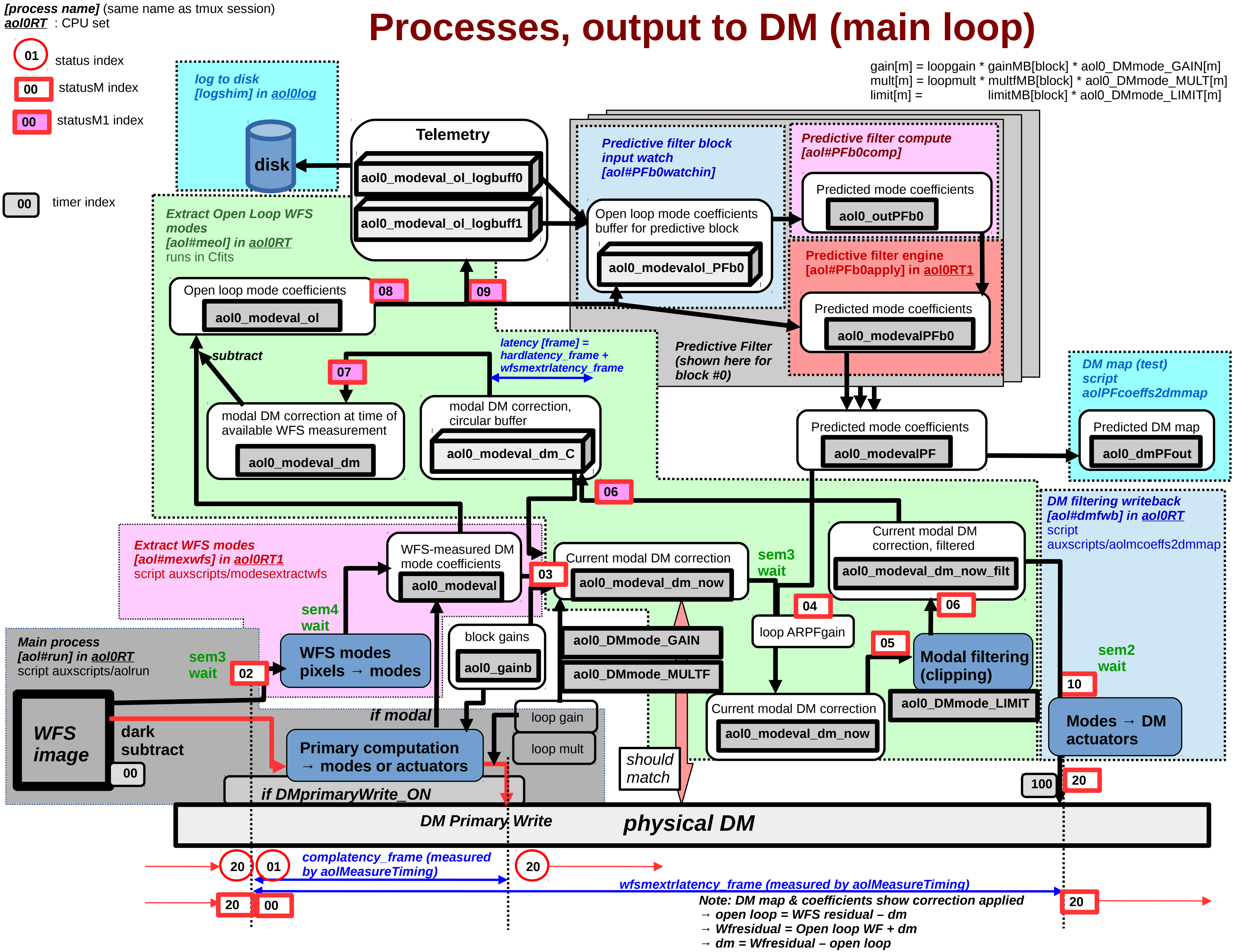

Making control modes (Zonal DM)
Create or load
dmmask
active DM actuators
Create or load
dmslaved
slaved DM actuators
Create
CPAmodes
Fourier modes
emodes
Excluded modes
zrespM
zonal RM
RMMmodes
low order modes
RMMresp
low order response
Create or load
wfsmask
active DM actuators
remove
extrapolate
3034
separate
fmodes0all
multiply
project
LOcoeff.txt
fmodesWFS00all
fmodes0_xx
fmodes2_xx
fmodes2all
remove null space within each block [SVDlim01]
SVDcoeff01_xx.txt
2056
remove null space within each block [SVDlim00]
remove DM modes contained in previous blocks, and enforce DM-space orthogonality between blocks [rmslim0]
fmodes2b_xx
fmodes2ball
fmodes1_xx
fmodes1all
2386
CREATE DM MODE BLOCKS
Modes are DM-orthogonal within and between blocks
ORTHOGONALIZE MODES IN WFS SPACE
multiply
project
fmodesWFS0_xx
fmodesWFS0all
remove WFS modes contained in previous blocks, and enforce WFS-space orthogonality between blocks [rmslim1]
fmodes3_xx
fmodes3all
fmodesWFS1_xx
fmodesWFS1all
SVDcoeff_xx.txt
remove WFS null space within each block [SVDlim]
(Modal DM)
zrespmat
DMmodes
fmodesWFS_xx
fmodesWFSall
SVD pseudo-inv
SVD pseudo-inv
cmatc_xx
cmatcact_xx
cmat.fits

OFFSETTING
LOWFS (loop #1) → PyWFS (loop #0)

Green color: process is part of loop #1



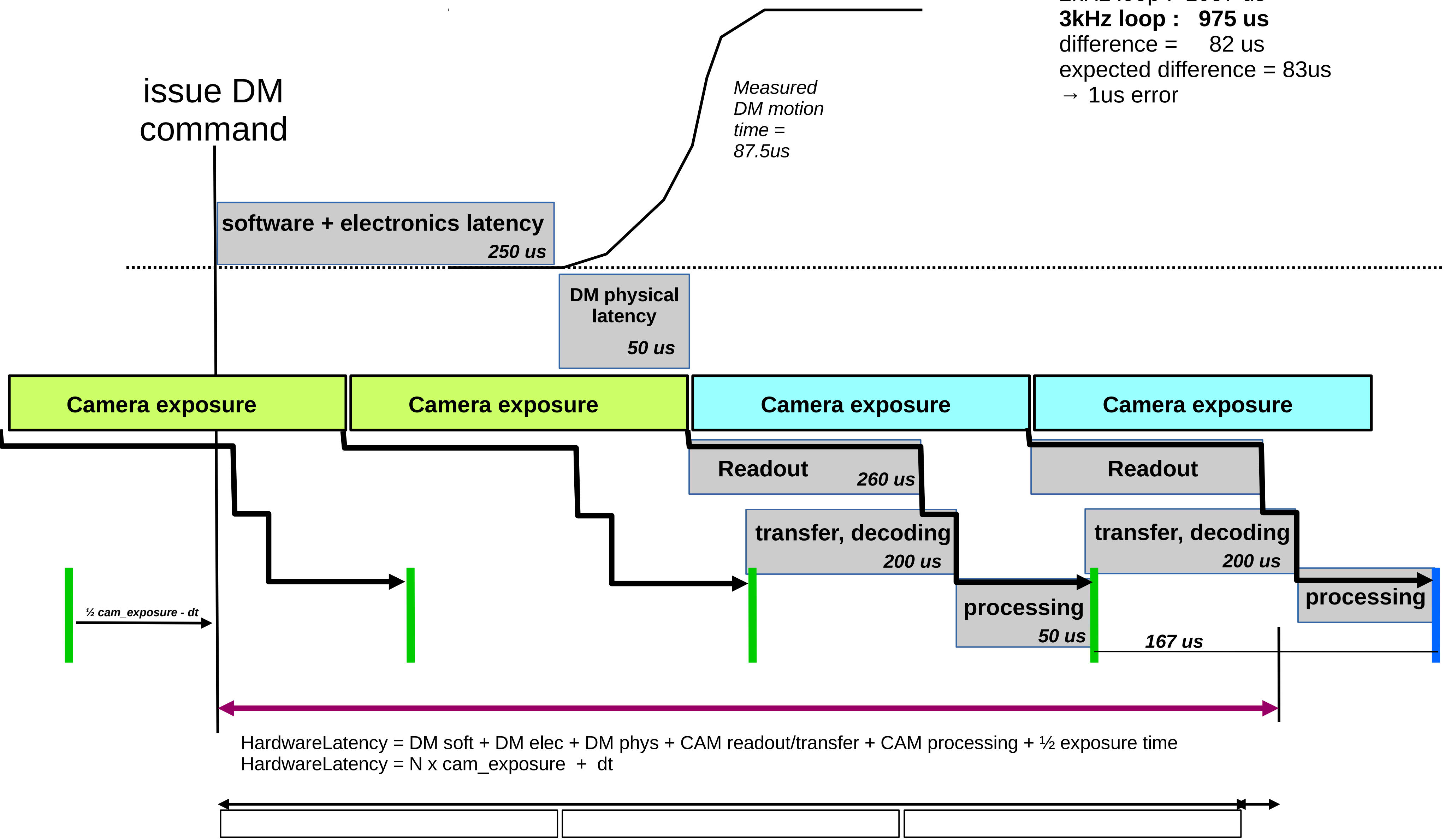
Processes, output to DM (main loop)



Hardware Latency

Definition:
*Time offset between **DM command issued**, and **mid-point between 2 consecutive WFS frames with largest difference***

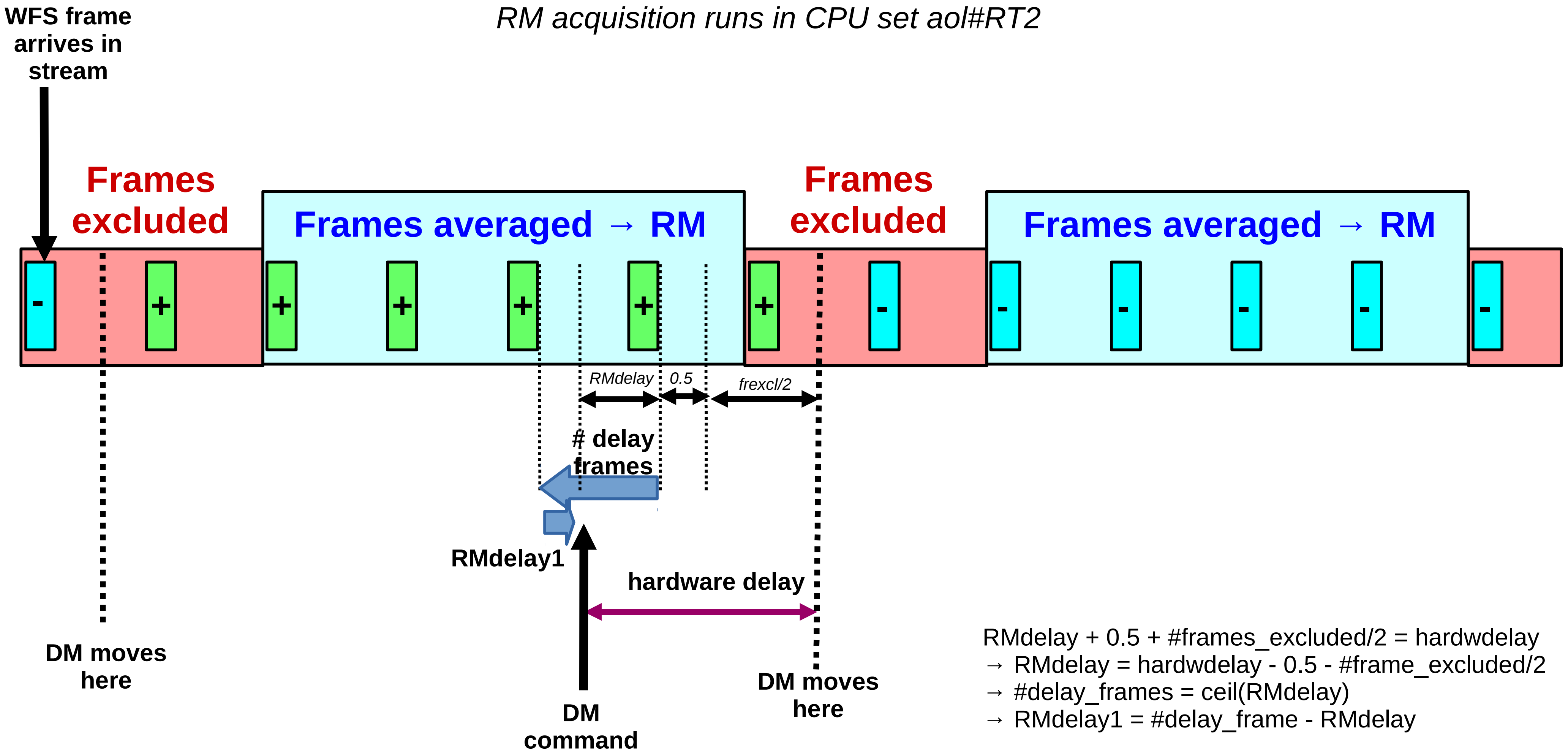
SCExAO :
2kHz loop : 1057 us
3kHz loop : 975 us
difference = 82 us
expected difference = 83us
→ 1us error



HardwareLatency = DM soft + DM elec + DM phys + CAM readout/transfer + CAM processing + ½ exposure time
HardwareLatency = N x cam_exposure + dt

RM acquisition - Timing

RM acquisition runs in CPU set aol#RT2

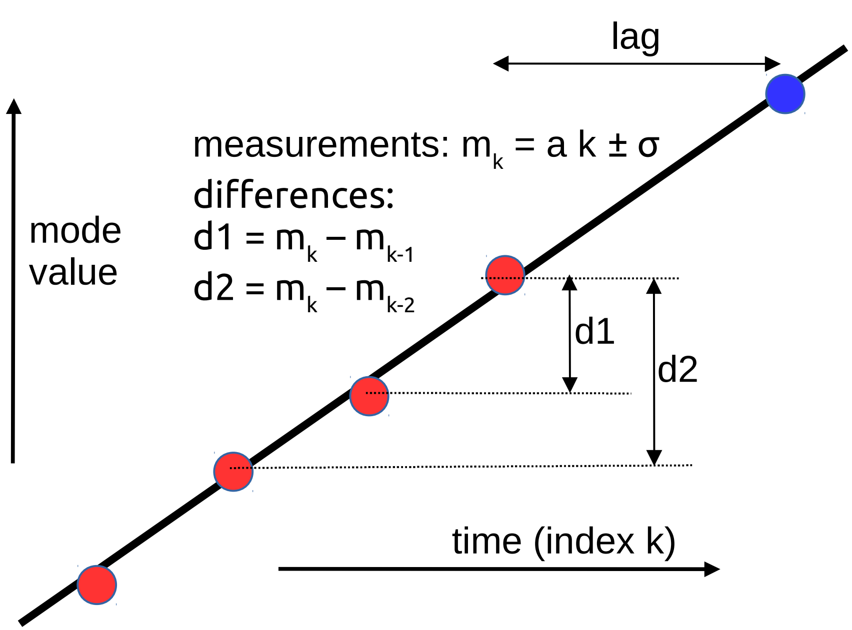


RMdelay + 0.5 + #frames_excluded/2 = hardwdelay
→ RMdelay = hardwdelay - 0.5 - #frame_excluded/2
→ #delay_frames = ceil(RMdelay)
→ RMdelay1 = #delay_frame - RMdelay

Automatic Gains Setting – Fast Mode

Goal: **Find optimal gain for each mode in non-predictive mode in bright star regime**. This mode should be very reactive and robust, and able to recompute 1200 optimal gains in < 300 us to allow gain updates @ up to 3 kHz.
Bright star regime: input WF mode evolves linearly with time (control frequency > vibrations)

→ Error is quadratic sum of time lag and measurement noise, which can be expressed as simple functions of recent measurements.



With integrator (gain = g)

Time lag error:
$\sigma_{TL} = a\,(\,lag + 1/g\,)$

Measurement noise propagation:
$\sigma_{MN} = \mathrm{sqrt}(g/(1-g))\ \sigma$

Estimating slope (a) and measurement noise (σ)

$\langle d1^2 \rangle = a^2 + 2\,\sigma^2$
$\langle d2^2 \rangle = 4\,a^2 + 2\,\sigma^2$

$\mathbf{a^2 = (\,\langle d2^2 \rangle - \langle d1^2 \rangle\,)\,/\,3}$
$\boldsymbol{\sigma^2 = (\,4\langle d1^2 \rangle - \langle d2^2 \rangle\,)\,/\,6}$

Real time process steps:
- Compute open loop coefficient mode values while loop is closed
- Update slope and measurement noise from running averages of $d1^2$ and $d2^2$
- Optimize $\sigma_{TL}^2 + \sigma_{MN}^2$ as a function of gain → update optimal gain