# EPIX10KA Data Access



## Calibration constants

- Jungfrau and Epix10ka Calibration

## Run, Step, and Event Loops

```
class Arguments:
  expt = 'ueddaq02'
  run = 66
  evtmax = 5
  detname = 'epixquad'

args = Arguments()

from psana.pyalgos.generic.NDArrUtils import info_ndarr
from psana import DataSource
ds = DataSource(exp=args.expt, run=args.run, dir=f'/cds/data/psdm/{args.expt[:3]}/{args.expt}/xtc')

for irun,run in enumerate(ds.runs()):
  print('\n==== %02d run: %d exp: %s detnames: %s' % (irun, run.runnum, run.expt, ','.join(run.detnames)))

  print('make %s detector object' % args.detname)
  det = run.Detector(args.detname)

  for istep,step in enumerate(run.steps()):
    print('\nStep %1d' % istep)

    for ievt,evt in enumerate(step.events()):
        if ievt>args.evtmax: exit('exit by number of events limit %d' % args.evtmax)

        print('%s\nEvent %04d' % (80*'_',ievt))
        segs = det.raw.segments(evt)
        raw = det.raw.raw(evt)

        print(info_ndarr(segs, 'segsments '))
        print(info_ndarr(raw, 'raw '))
```

**output of the above script**

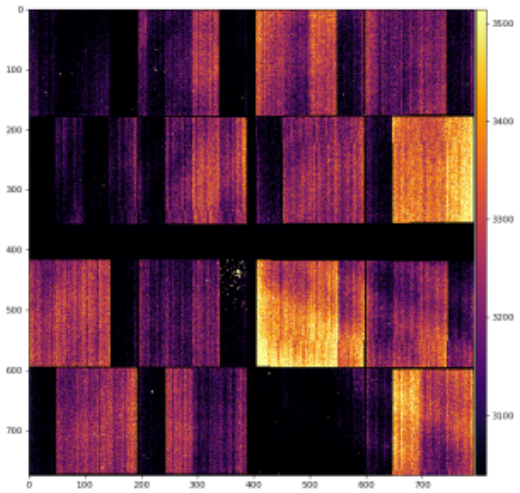
```
==== 00 run: 66 exp: ueddaq02 detnames: timing,epixquad
make epixquad detector object

Step 0
_____________________________________________________________________________
Event 0000
segsments shape:(4,) size:4 dtype:uint16 [0 1 2 3]
raw shape:(4, 352, 384) size:540672 dtype:uint16 [3304 3385 3401 3407 3291...]
_____________________________________________________________________________
Event 0001
segsments shape:(4,) size:4 dtype:uint16 [0 1 2 3]
raw shape:(4, 352, 384) size:540672 dtype:uint16 [3278 3381 3385 3393 3298...]
_____________________________________________________________________________
Event 0002
segsments shape:(4,) size:4 dtype:uint16 [0 1 2 3]
raw shape:(4, 352, 384) size:540672 dtype:uint16 [3317 3431 3431 5256 5790...]
_____________________________________________________________________________
Event 0003
segsments shape:(4,) size:4 dtype:uint16 [0 1 2 3]
raw shape:(4, 352, 384) size:540672 dtype:uint16 [3325 3439 3426 3462 3361...]
_____________________________________________________________________________
Event 0004
segsments shape:(4,) size:4 dtype:uint16 [0 1 2 3]
raw shape:(4, 352, 384) size:540672 dtype:uint16 [3328 3436 3434 3456 3348...]
_____________________________________________________________________________
Event 0005
segsments shape:(4,) size:4 dtype:uint16 [0 1 2 3]
raw shape:(4, 352, 384) size:540672 dtype:uint16 [3335 3428 3426 3454 3347...]
exit by number of events limit 5
```

# Methods of det.raw

## Methods for AMI interface

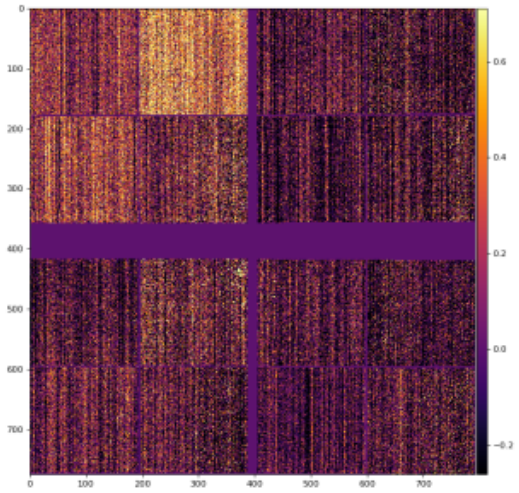
### raw

raw = det.raw.raw(evt)




### calib

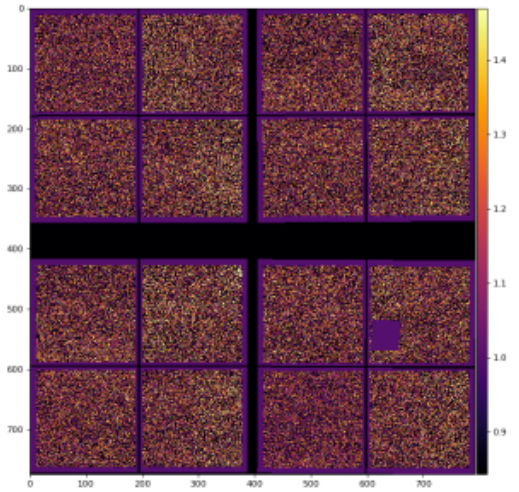
clb = det.raw.calib(evt)



user_mask = np.ones_like(det.raw.raw(evt), dtype=DTYPE_MASK) #np.uint8
user_mask[0,100:150,200:250] = 0

clb = det.raw.calib(evt, cmpars=(7,2,100,10), mbits=0o7, mask=user_mask, edge_rows=10, edge_cols=10, center_rows=5, center_cols=5, **kwargs)

common mode correction, user defined mask, status mask and mask of edges are applied:



### image

img = det.raw.image(evt, **kwargs)

img = det.raw.image(evt)

img = det.raw.image(evt, nda=arr, pix_scale_size_um=args.pscsize, mapmode=args.mapmode)

### segments

segs = det.raw.segments(evt)

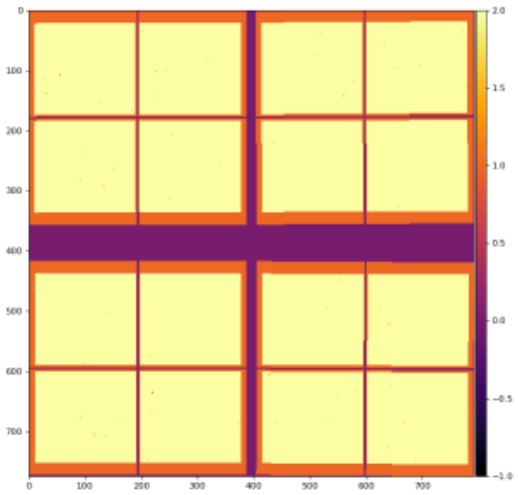
## Other useful methods hidden for AMI

### _mask and _mask_comb

msk = det.raw._mask(calib=False, status=False, edges=False, neighbors=False, **kwargs)

msk = det.raw._mask_comb(mbits=0o7, mask=None, edge_rows=10, edge_cols=10, center_rows=5, center_cols=5)

where mbits &1 - calib, &2 - status,  &4 -edges , &8 - neighbors



### Separate masks

o = det.raw

m = o._mask_default(dtype=DTYPE_MASK)
m = o._mask_calib_or_default(dtype=DTYPE_MASK)
m = o._mask_from_status(**kwa)
m = o._mask_edges(edge_rows=10, edge_cols=10, center_rows=5, center_cols=5, **kwa)

### Pixel coordinate arrays

a = o._pixel_coord_indexes(**kwa) # 'pix_scale_size_um',None; 'xy0_off_pix',None; # do_tilt',True; 'cframe',0
a = o._pixel_coords(**kwa) # do_tilt',True; 'cframe',0

# References


- EPIX10KA2M References
- Jungfrau and Epix10ka Calibration
- Common mode correction algorithms
- https://github.com/slac-lcls/lcls2/blob/master/psana/psana/detector/areadetector.py