

# Andor image processing

## Content

- [Content](#)
- [Setup environment](#)
- [Save image in TIFF file](#)
  - [Configuration file](#)
    - [Configuration file for Python module pyimgalgos.image\\_save\\_in\\_file](#)
    - [Configuration file for C++ module ImgAlgos.ImgSaveInFile](#)
  - [Get TIFF files from data](#)
- [Andor calibrated image](#)
  - [Calibration constants](#)
  - [Setup environment for current release ana-0.13.17](#)
  - [Get calibrated image](#)
  - [Image examples](#)
    - [Raw image full-size and zoomed](#)
    - [Image with subtracted pedestals](#)
    - [Image with subtracted common mode](#)
    - [pedestals - Image averaged by calibman](#)
- [References](#)

## Setup environment

```
ssh psana  
  
cd <your-directory>  
  
sit_setup
```

## Save image in TIFF file

### Configuration file

There are two versions of image saving modules `ImgAlgos.ImgSaveInFile` (C++) and `pyimgalgos.image_save_in_file` (Python), which create slightly different TIFF files.

### Configuration file for Python module `pyimgalgos.image_save_in_file`

Content of the file [psana-mecb3114-r0017-tiff.cfg](#)

```
[psana]  
files = exp=mecb3114:run=17  
#events = 5  
#skip-events = 0  
  
modules = ImgAlgos.AndorImageProducer \  
          pyimgalgos.image_save_in_file  
  
[ImgAlgos.AndorImageProducer]  
source = DetInfo(MecTargetChamber.0:Andor.1)  
key_in =  
key_out = andor_img  
outtype = asdata  
print_bits = 1  
  
[pyimgalgos.image_save_in_file]  
source = DetInfo(MecTargetChamber.0:Andor.1)  
key_in = andor_img  
#ofname = andor.1.txt  
ofname = andor.1.tiff  
print_bits = 5
```

## Configuration file for C++ module `ImgAlgos.ImgSaveInFile`

Module `ImgAlgos.ImgSaveInFile` can be added to the configuration file as:

```
modules = ImgAlgos.AndorImageProducer \
...
        ImgAlgos.ImgSaveInFile
...

[ImgAlgos.ImgSaveInFile]
source      = DetInfo(MecTargetChamber.0:Andor.1)
key         = andor_img
saveAll     = yes
#ftype     = tiff
ftype      = txt
fname      = andor
print_bits  = 3
```

## Get TIFF files from data

Run `psana` with configuration file and optional experiment/run number structure `exp=mecb3114:run=17` by the command:

```
psana -c psana-mecb3114-r0017-tiff.cfg exp=mecb3114:run=17
```

Tiff files will be produced with common prefix name `andor.1` (which may contain path to the directory as well):

```
andor.1-mecb3114-r0017-ev000001.tiff
andor.1-mecb3114-r0017-ev000002.tiff
andor.1-mecb3114-r0017-ev000003.tiff
```

## Andor calibrated image

### Calibration constants

Dark run calibration can be done with procedure [calibman](#) (since ana-0.13.18). Files can be deployed in the expected place, for example

`/reg/d/psdm/SXR/sxrg3715/calib/Andor::CalibV1/SxrEndstation.0:Andor.2/pedestals/0-end.data`

for types `pedestals pixel_rms pixel_status`.



Runs of `sxrg3715` have image every 10 sec it means that 1199 events are empty. In order to get calibration files the `calibman` default parameters needs to be changed.

Start `calibman`, click on tab "Configuration" then on tab "Parameters" and set field "end:" to 100000 and field "scan" to "2000". After that regular dark run calibration should work. Do not forget to coise the detector ANDOR.

The same trick with event numbers can be acheived in the `calibrun` command skeeping empty 1195 events:

```
calibrun -e sxrg3715 -d ANDOR -c ./calib -P -D -r 46 -n 100000 -s 1195
```

or

```
calibrun -e sxrg3715 -d ANDOR -P -D -r 46 -n 100000 -s 1195
```

or

```
calibrun -e sxrg3715 -d ANDOR -P -D -r 46 -n 100000 -m 2000
```

Type common\_mode constants can be copied from file with parameters for mode 2 (see : [Common mode correction algorithms](#) ), for example

```
2 20 20 512 0 0 0 0 0 0 0 0 0 0 0
```

## Setup environment for current release ana-0.13.17

```
ssh psana  
  
cd <your-any-directory>  
  
newrel ana-current my-local-release-dir;  
cd my-local-release-dir;  
sit_setup;  
addpkg CalibManager HEAD;  
scons;
```

At this point your local release is built and regular session looks like:

```
cd my-local-release-dir;  
sit_setup;  
calibman;
```

OR: use command line calibrun in stead of GUI-based calibman with deployment of files in local or standard place:

```
calibrun -e sxrg3715 -c ./calib -d ANDOR -P -D -r 46 -n 100000 -s 1195;  
calibrun -e sxrg3715 -d ANDOR -P -D -r 46 -n 100000 -s 1195;  
calibrun -e sxrg3715 -d ANDOR -P -D -r 46 -n 100000 -m 2000;
```

## Get calibrated image

In psana Andor image can be retrieved from raw data to ndarray using module `ImgAlgos.AndorImageProducer` . Then it can be calibrated by the module

`ImgAlgos.NDArrCalib` and further processed in python or saved as in the example.

Example of the configuration file `psana-sxrg3715-r0046-andor-image.cfg` :

```

[psana]
# Default calibration directory:
# calib-dir = /reg/d/psdm/sxr/sxrg3715/calib
calib-dir = /reg/neh/home1/dubrovin/LCLS/PSANA-V01/test-sxrg3715/calib

files = exp=sxrg3715:run=46
#events = 5
#skip-events = 0

modules = ImgAlgos.AndorImageProducer:a2 \
          ImgAlgos.NDArrCalib:a2 \
          ImgAlgos.ImgSaveInFile:a2

[ImgAlgos.AndorImageProducer:a2]
source = DetInfo(SxrEndstation.0:Andor.2)
key_in =
key_out = andor_img_raw
#outtype = asdata
outtype = float
print_bits = 9

[ImgAlgos.NDArrCalib:a2]
source = DetInfo(SxrEndstation.0:Andor.2)
key_in = andor_img_raw
key_out = andor_img_clb
outtype = float
do_peds = yes
do_cmod = yes
print_bits = 15

[ImgAlgos.ImgSaveInFile:a2]
source = DetInfo(SxrEndstation.0:Andor.2)
key = andor_img_clb
saveAll = yes
#ftype = tiff
ftype = txt
fname = andor-2-cm2
print_bits = 3

```

which can be executed by the command:

```
psana -c psana-sxrg3715-r0046-andor-image.cfg
```

## Image examples

Event 12000, exp=sxrg3715:run=46, SxrEndstation.0:Andor.2

### Raw image full-size and zoomed

### Image with subtracted pedestals

evaluated using the same run 46

### Image with subtracted common mode

[calib/Andor::CalibV1/SxrEndstation.0:Andor.2/common\\_mode/0-end.data](#)

```
2 20 20 512 0 0 0 0 0 0 0 0 0 0 0 0
```

This algorithm does not improve image quality. Close consideration shows that the mean intensity value of each row of pixels is consistent with zero. The RMS spread is low for "green" horizontal stripes and larger for "color-full" stripes.

## **pedestals - Image averaged by calibman**

Average over 10 events of exp=xrg3715:run=46, SxrEndstation.0:Andor.2

Horizontal stripes almost disappear on averaged image. That confirms, that mean offset of each stripe is about zero.

## **References**

- [Psana Module Examples](#)
- [Psana Module Catalog](#)
- [Calibration Management Tool](#)
- [Common mode correction algorithms](#)