VERA C. RUBIN OBSERVATORY

Calibration Production

A Short Introduction

2024-04-02

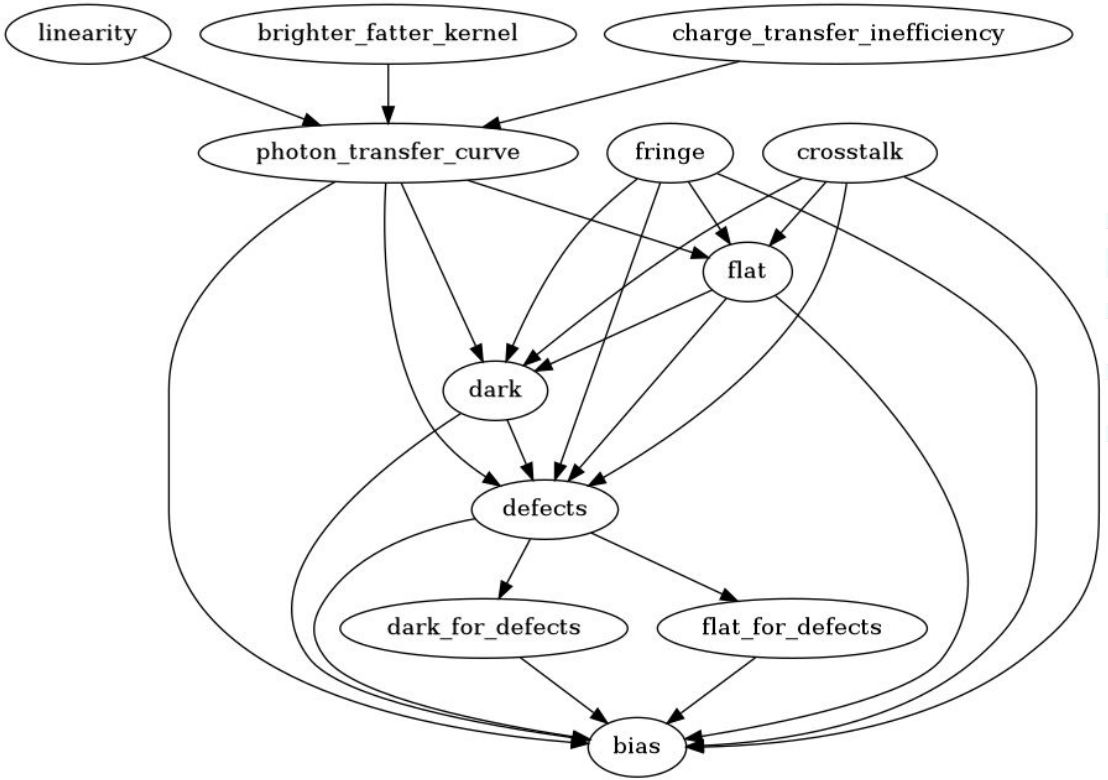
NSF AURA U.S. DEPARTMENT OF ENERGY SLAC CHARLES AND LISA SIMONYI FUND FOR ARTS AND SCIENCES LSST CORPORATION

How do we make calibrations?

- Identify input exposures.
- Run the cp_pipe pipeline for that calibration.
- Run the cp_verify pipeline for that calibration.
- Get approval from the TAXICAB.
- Certify calibrations.
- Transfer calibrations to required repos.
- Ensure collection chains are correct.

Prerequisites (and what each calib does):

- bias: electronics, f(t=0)
- dark: electronics, f(t)
- flat: illumination/QE/gain
  - This will be changing.
- defects: bad pixels, hot/cold
- PTC: find gain/RN/covar
- fringe: long wavelength light
- crosstalk: signal transfer
- linearity: is bright bright?
- bfk: if it's too bright, it grows
- cti: how much charge sticks



Identify input exposures:

- There is no perfect algorithm for this, and we assume that the verification process will identify any problems.
- Best practices:
  - Never include the first exposure in a given set.
    - The camera doesn't fully flush the pixels/readout/etc. prior to an exposure.
    - This means there's an unknown amount of dark signal that we won't be able to correct.
  - Usually a single set will be sufficient.
    - N ~ 20 is generally my target.
  - A single set of flats should have consistent illumination.
- We also need to choose verification exposures:
  - I usually throw in the first exposure here.
  - Any other appropriate inputs from that day.
  - Choose some from other dates to look at stability.
  - A sampling is usually fine (every 3rd/5th/10th exposure from many dates/sets).
- Consistent camera state:
  - `SEQNAME, SEQFILE, ODP, AP0_RC, TEMP_SET,` etc.

Querying for exposures:

- Via the butler:
  - `butler query-dimension-records /repo/embargo exposure –where “instrument='LATISS'”`
  - This is often all that’s needed.
  - Especially if a special calibration observing day was taken (2024-03-26 for example).
- Via side-band SQL database:
  - `sqlite3 ~czw/dev/tools/butler.db “select id,physical_filter,exposure_time, observation_type,observation_reason,target_name from exposure WHERE day_obs=20240326”`
  - This is useful if we need to find dates with sufficient exposures with particular qualities.
- What observation_types do we need?
  - bias → bias
  - dark → dark
  - flat, ptc, cti → flat
  - defects → combined bias, combined dark, combined flat
  - fringe → science exposures, y-band (z-band?)
  - crosstalk → science exposures with bright stars

Querying for exposures:

xterm

```
id      physical_filter exposure_time  observation_type       observation_reason      target_name
2024032600001   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600002   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600003   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600004   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600005   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600006   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600007   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600008   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600009   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600010   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600011   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600012   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600013   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600014   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600015   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600016   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600017   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600018   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600019   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600020   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600021   SDSSr_65mm~empty 0.0     bias    bias    FlatField position
2024032600022   SDSSr_65mm~empty 5.0     dark    dark    FlatField position
2024032600023   SDSSr_65mm~empty 5.0     dark    dark    FlatField position
2024032600024   SDSSr_65mm~empty 5.0     dark    dark    FlatField position
2024032600025   SDSSr_65mm~empty 5.0     dark    dark    FlatField position
2024032600026   SDSSr_65mm~empty 5.0     dark    dark    FlatField position
2024032600027   SDSSr_65mm~empty 15.0    dark    dark    FlatField position
2024032600028   SDSSr_65mm~empty 15.0    dark    dark    FlatField position
2024032600029   SDSSr_65mm~empty 30.0    dark    dark    FlatField position
2024032600030   SDSSr_65mm~empty 30.0    dark    dark    FlatField position
```

xterm

```
instrument      id          physical_filter            obs_id       exposure_time dark_time  ob
servation_type                  observation_reason                    day_obs  seq_num seq_st
art seq_end        group_name          group_id           target_name              science_
program           tracking_ra         tracking_dec        sky_angle               azimuth
      zenith_angle    has_simulated             timespan (TAI)
---------- ------------ ------------------------ ------------------- ------------- ---------- --
-------------- ------------------------------------------------------------ -------- ------- ------
--- ------- ------------------------ ------------------- ---------------------- ---------- -------------
------------- ---------------------- --------------------- ---------------------- --------------------
--- ------------------ ------------ ----------------------------------------
    LATISS 2022083100004          unknown~unknown AT_O_20220831_000004          0.0  0.0160482
         bias                                                          bias 20220831      4
 4       4            2022083100004       2022083100004                  UNKNOWN
    unknown                  None                None                    None                  N
one             None        False [2022-08-31T22:58:13, 2022-08-31T22:58:13)
    LATISS 2022083100005          unknown~unknown AT_O_20220831_000005          0.0   0.011363
         bias                                                          bias 20220831      5
 5       5            2022083100005       2022083100005                  UNKNOWN
    unknown                  None                None                    None                  N
one             None        False [2022-08-31T22:59:03, 2022-08-31T22:59:03)
    LATISS 2022083100006          unknown~unknown AT_O_20220831_000006          0.0  0.0149965
         bias                                                          bias 20220831      6
 6       6            2022083100006       2022083100006                  UNKNOWN
    unknown                  None                None                    None                  N
one             None        False [2022-08-31T22:59:07, 2022-08-31T22:59:07)
    LATISS 2022083100007          unknown~unknown AT_O_20220831_000007          0.0  0.0119367
         bias                                                          bias 20220831      7
 7       7            2022083100007       2022083100007                  UNKNOWN
    unknown                  None                None                    None                  N
one             None        False [2022-08-31T22:59:11, 2022-08-31T22:59:11)
    LATISS 2022083100008          unknown~unknown AT_O_20220831_000008          0.0  0.0111244
```

- fringe → science exposures, y-band (z-band?)
- crosstalk → science exposures with bright stars

Running the pipelines:

- Follow the example script (`~czw/dev/calibConstruction/templates/example.sh`):
  - I've avoided BPS because it's not essential for LATISS and ComCam, but will be needed for LSSTCam.
  - I'm open to any changes that can speed things up/make it easier to understand, as long as:
  - **Every step gets recorded.**
    - Knowing what we did, even if we did it wrong, is the most important thing.
    - Plus, if you did it right, you can just copy the command the next time.

```
#
#
# Example command list for generating calibrations.
# THIS SOULD NOT BE DIRECTLY RUN
#
#
exit 1
kill -9 $$

# Almost all calibrations will be built in the "embargo" repo.
# TICKET is the ticket covering this work.
# TAG    is a human readable name so it's clear what these calibs are for.
# RERUN  is a "20240209a" like string indicating when the calibs were made.
REPO=/repo/embargo
TICKET=DM-43525
TAG=lowerSetTemp
RERUN=20240326a

#
#
#
#
# bias
EXPOSURES=2024032600002..2024032600021
BIAS_V=2024032600001,2024032600066..2024032600069,2024032600095..2024032600098,2024032600124..2024032600127,2024032600153..2024032600156
pipetask --long-log run -b $REPO \
     -p $CP_PIPE_DIR/pipelines/Latiss/cpBias.yaml \
     -i LATISS/raw/all,LATISS/calib \
     -o LATISS/calib/$TICKET/$TAG/biasGen.$RERUN \
     -d "instrument='LATISS' AND detector=0 AND exposure IN ($EXPOSURES)" \
     -j 4 |& tee biasGen.$RERUN.log

butler query-collections /repo/embargo LATISS/calib/$TICKET/$TAG/biasGen.$RERUN

BIAS_RUN=LATISS/calib/DM-43525/lowerSetTemp/biasGen.20240326a/20240327T163518Z
```

Running the pipelines:

- Follow the example script (`~czw/dev/calibConstruction/templates/example.sh`):
  - I've avoided BPS because it's not essential for LATISS and ComCam, but will be needed for LSSTCam.
  - I'm open to any changes that can speed things up/make it easier to understand, as long as:
  - **Every step gets recorded.**
    - Knowing what we did, even if we did it wrong, is the most important thing.
    - Plus, if you did it right, you can just copy the command the next time.
- Verification is included in the example script as well.
  - Use the exposures that went into the calibration (internal verification: do all inputs agree?)
  - Add additional exposures that were not included (external verification: is the calib consistent in time?)
  - papermill can pre-process a notebook, setting parameterized variables (current visualization).
  - analysis_tools metrics are coming, slowly (input data should exist when run with DM-42927).

Running the pipelines:

```
# Verification runs.
GEN_RUNS=$BIAS_RUN,$DARK_RUN,$DEFECTS_RUN,$FLAT_G_RUN,$FLAT_R_RUN,$FLAT_Z_RUN,$FLAT_Y_RUN,$FLAT_WHITE_RUN,$PTC_RUN
# $FLAT_I_RUN,

# Use in development DM-42927 versions of analysis_tools and
# cp_verify.  This should provide new datasets that can be used to
# test out the analysis_tools work.  A compromise to get calibs out
# while still transitioning to the new verification procedure.
#
#
#
#
# bias verification
EXPOSURES=2024032600002..2024032600021
EXPOSURES_V=2024032600001,2024032600066..2024032600069,2024032600095..2024032600098,2024032600124..2024032600127,2024032600153..2024032600156

pipetask --long-log run -b $REPO \
     -p $CP_VERIFY_DIR/pipelines/Latiss/VerifyBias.yaml \
     -i $GEN_RUNS,LATISS/calib,LATISS/raw/all \
     -o LATISS/calib/$TICKET/$TAG/verifyBias.$RERUN \
     -d "instrument='LATISS' AND detector=0 AND exposure IN ($EXPOSURES, $EXPOSURES_V)" \
     --register-dataset-types \
     -j 4 |& tee verifyBias.$RERUN.log
papermill --prepare-only -p interactive False -p repository $REPO \
          -p genCollection LATISS/calib/$TICKET/$TAG/biasGen.$RERUN \
          -p verifyCollection LATISS/calib/$TICKET/$TAG/verifyBias.$RERUN \
          $CP_VERIFY_DIR/notebooks/cpVerifyBias.ipynb ./cpVerifyBias-$RERUN.ipynb
```

TAXICAB approval:

- TBD.
- First meeting is 2024-04-11.

Certify the calibrations:

- Choose the start date (and for historical data, the end date) for calib validity.
- Certify the calibration to a new collection:
  - A CALIBRATION collection is like a symlink to the calibration in its generation RUN collection.
  - But with the validity date range added as an extra dimension.
  - You can always query the calibration in its RUN collection without a date.
    - This is how we ran the verification.
  - But through the calibration collection, you need to specify an exposure value for the date lookup.
  - Butler collections are searched sequentially for the first match.
- Create a collection chain for the calibrations created on the ticket.
- Prepend that chain to the start of the top-level collection.
- Export the calibrations

Certif

- Choose the start … alib validity.
- Certify the calib…
  - A CALIBRATION … RUN collection.
  - But with the va…
  - You can always…
    - This is h…
  - But through th… for the date lookup.
  - Butler collecti…
- Create a collecti… t.
- Prepend that ch…
- Export the calibr…

```
# Almost all calibrations will be built in the "embargo" repo.
# TICKET is the ticket covering this work.
# TAG    is a human readable name so it's clear what these calibs are for.
# RERUN  is a "20240209a" like string indicating when the calibs were made.
# START_DATE is the date we will start using these calibs
# CERT_RERUN is the RERUN string for the certification date.
REPO=/repo/embargo
TICKET=DM-43525
TAG=lowerSetTemp
RERUN=20240326a
START_DATE=2024-04-10
CERT_RERUN=20240405a
YRERUN=20240328a

#
#
# Certify commands:
butler certify-calibrations --begin-date $START_DATE $REPO \
    LATISS/calib/$TICKET/$TAG/biasGen,$RERUN \
    LATISS/calib/$TICKET/$TAG/bias,$CERT_RERUN \
    bias
butler certify-calibrations --begin-date $START_DATE $REPO \
    LATISS/calib/$TICKET/$TAG/darkGen,$RERUN \
    LATISS/calib/$TICKET/$TAG/dark,$CERT_RERUN \
    dark
butler certify-calibrations --begin-date $START_DATE $REPO \
    LATISS/calib/$TICKET/$TAG/defectGen,$RERUN \
    LATISS/calib/$TICKET/$TAG/defects,$CERT_RERUN \
    defects
butler certify-calibrations --begin-date $START_DATE $REPO \
    LATISS/calib/$TICKET/$TAG/flatGen,g,$RERUN \
    LATISS/calib/$TICKET/$TAG/flat-g,$CERT_RERUN \
    flat
```

Certify the calibrations:

- Choose t… start … …alidity.
- Certify the calibra…
  - A CALIBRATION … …n RUN …tion.
  - But with the val…
  - You can always …
    - This is ho…
  - But through the… …e for the date lookup.
  - Butler collectio…
- Create a collectio… …et.
- Prepend that cha…
- Export the calibra…

```
# Pre-chain:
#   This is here so we know what state things were in before doing anything.
butler query-collections $REPO LATISS/calib

butler collection-chain $REPO \
    LATISS/calib/$TICKET \
    LATISS/calib/$TICKET/$TAG/bias,$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/dark,$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/defects,$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/flat-g,$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/flat-r,$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/flat-i,$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/flat-z,$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/flat-y,$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/flat-white,$CERT_RERUN

butler collection-chain $REPO \
    --mode prepend \
    LATISS/calib \
    LATISS/calib/$TICKET

# Post-chain:
#   This is here so we know what state things were in when we finished.
butler query-collections $REPO LATISS/calib

# Export the new calibrations:
butler export-calibs $REPO \
    --transfer copy \
    ./calibs,$CERT_RERUN \
    LATISS/calib/$TICKET \
```

Transfer to other repos:

- This is easy for `/repo/main`.

Tran

- This is easy for

```
#
#
#
# Switch repo:
REPO=/repo/main

# Pre-chain:
#   This is here so we know what state things were in before doing anything.
butler query-collections $REPO LATISS/calib

butler import $REPO ./calibs.$INSTALL_RERUN \
    --transfer copy \
    --export-file ./calibs.$INSTALL_RERUN/export.yaml \
    -s instrument -s detector -s physical_filter

butler collection-chain $REPO \
    LATISS/calib/$TICKET \
    LATISS/calib/$TICKET/$TAG/bias.$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/dark.$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/defects.$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/flat-g.$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/flat-r.$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/flat-i.$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/flat-z.$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/flat-y.$CERT_RERUN \
    LATISS/calib/$TICKET/$TAG/flat-white.$CERT_RERUN

# chmod:
# Needed if we don't chmod the export:
REFERENCE=/sdf/group/rubin/repo/main/LATISS/calib/DM-36719/flatGen-SDSSr.20221107a/20
ty_RXX_S00_LATISS_calib_DM-36719_flatGen-SDSSr_20221107a_20221107T235401Z.fits
TARGETS=`find /sdf/group/rubin/repo/main/LATISS/calib/$TICKET/ -iname '*.fits'`
chmod --reference=$REFERENCE $TARGETS
```

Transfer to other repos:

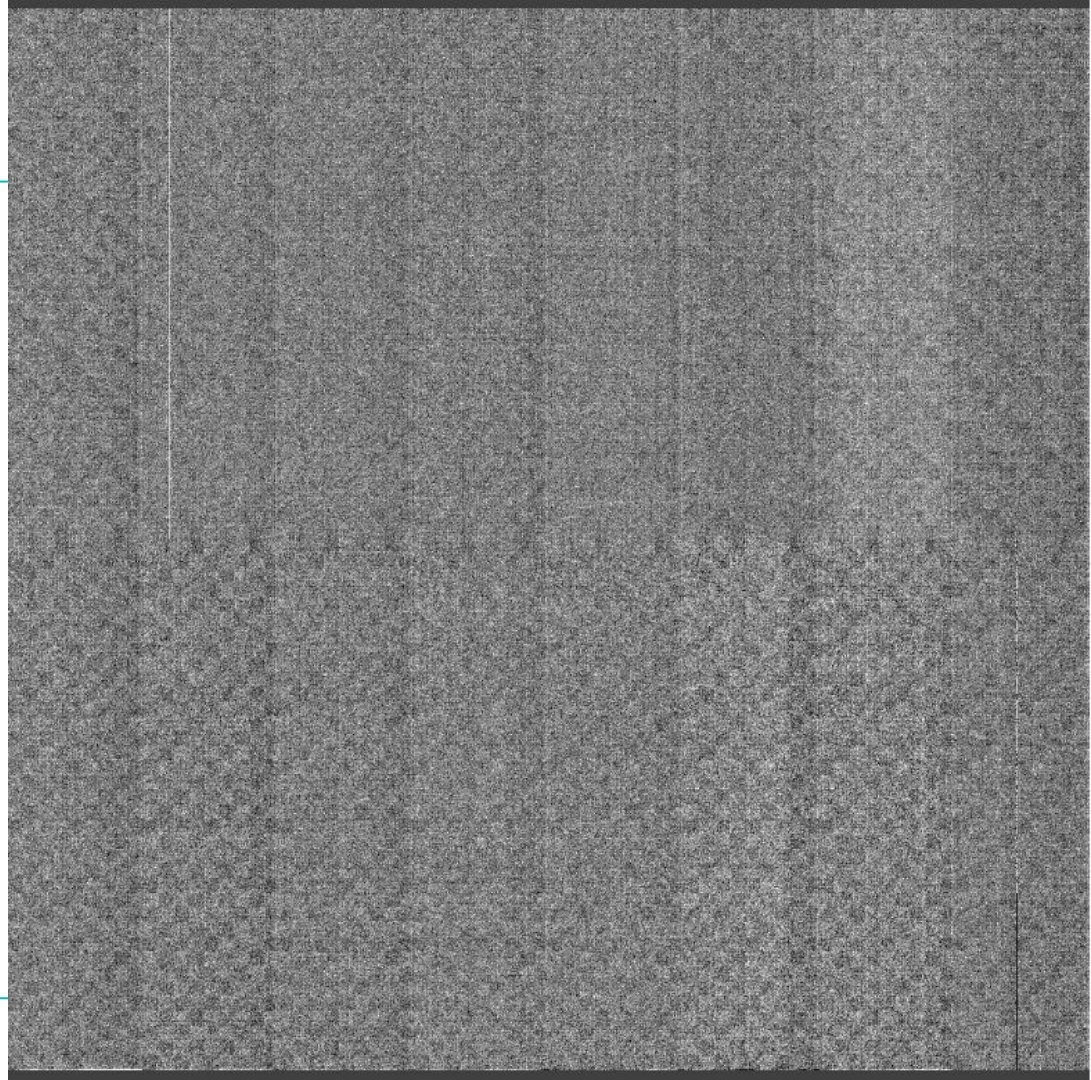
- This is easy for `/repo/main`.
- But we also need new calibrations at the summit (`/repo/LATISS` or `/repo/LSSTComCam`).
- TBD: transfer to the summit:
  - It will be something like `rsync ./calibs.$CERT_RERUN nfs1.cp.lsst.org:/some/temp/path`
  - https://rubinobs.atlassian.net/browse/IHS-7855
- This also requires access to the summit:
  - I plan to do a mass request to get everyone access.
  - There is a VPN. It is a bit of a process to get connected.
  - There is also a summit ownership group to control file access.
- Check that final collections match in all locations.
- Ensure files are chmod-ed correctly.

Demos?

- I would like to have a demo notebook like we have for DP0.
- In the meantime, the ci_cpp can serve as a test case.
  - It's what I'm using for the analysis_tools development
- Get the packages:
  - `git clone github.com:lsst/testdata_latiss_cpp`
    - This is a git-lfs backed repo containing a complete set of input exposures.
  - `git clone github.com:lsst/ci_cpp_gen3`
    - This holds all the code. It will one day be renamed (as the Gen3 middleware is ubiquitous).
- Set them up:
  - `setup -j -r ./testdata_latiss_cpp && setup -j -r ./ci_cpp_gen3`
- Run the whole thing:
  - `cd ci_cpp_gen3 && scons -c && scons -j $N_JOBS`
    - This must be run from scratch each time, so you need to clean with `-c`.
- Or just get the commands that will happen:
  - `cd ci_cpp_gen3 && mkdir ./.scons_temp && scons -n > ../ci_cpp_gen3.cmds`
    - `scons` cannot list commands without a temp directory, so this makes it if it doesn't yet exist.

Examples:

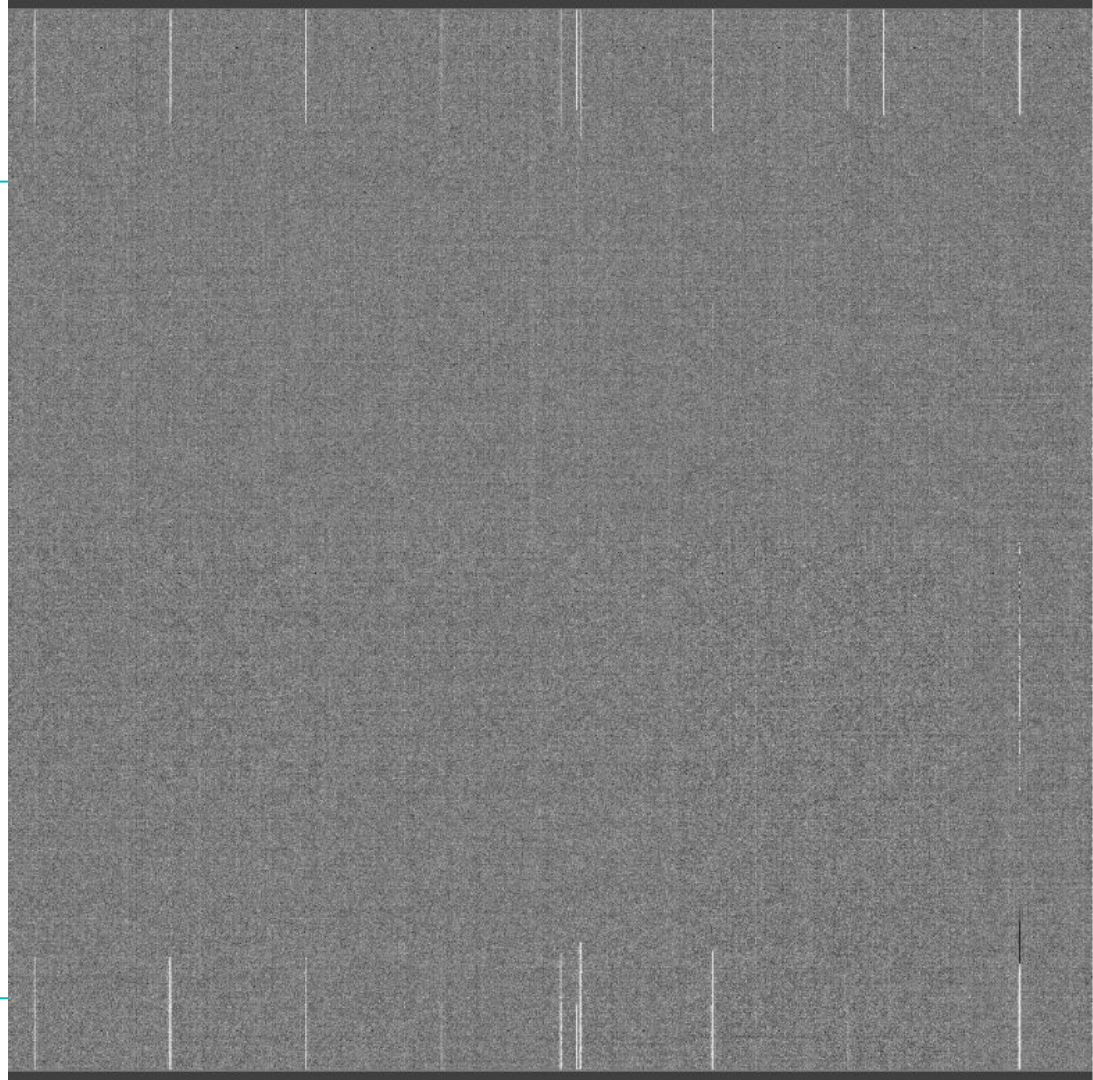
Combined Bias



Examples:

Residual Bias:

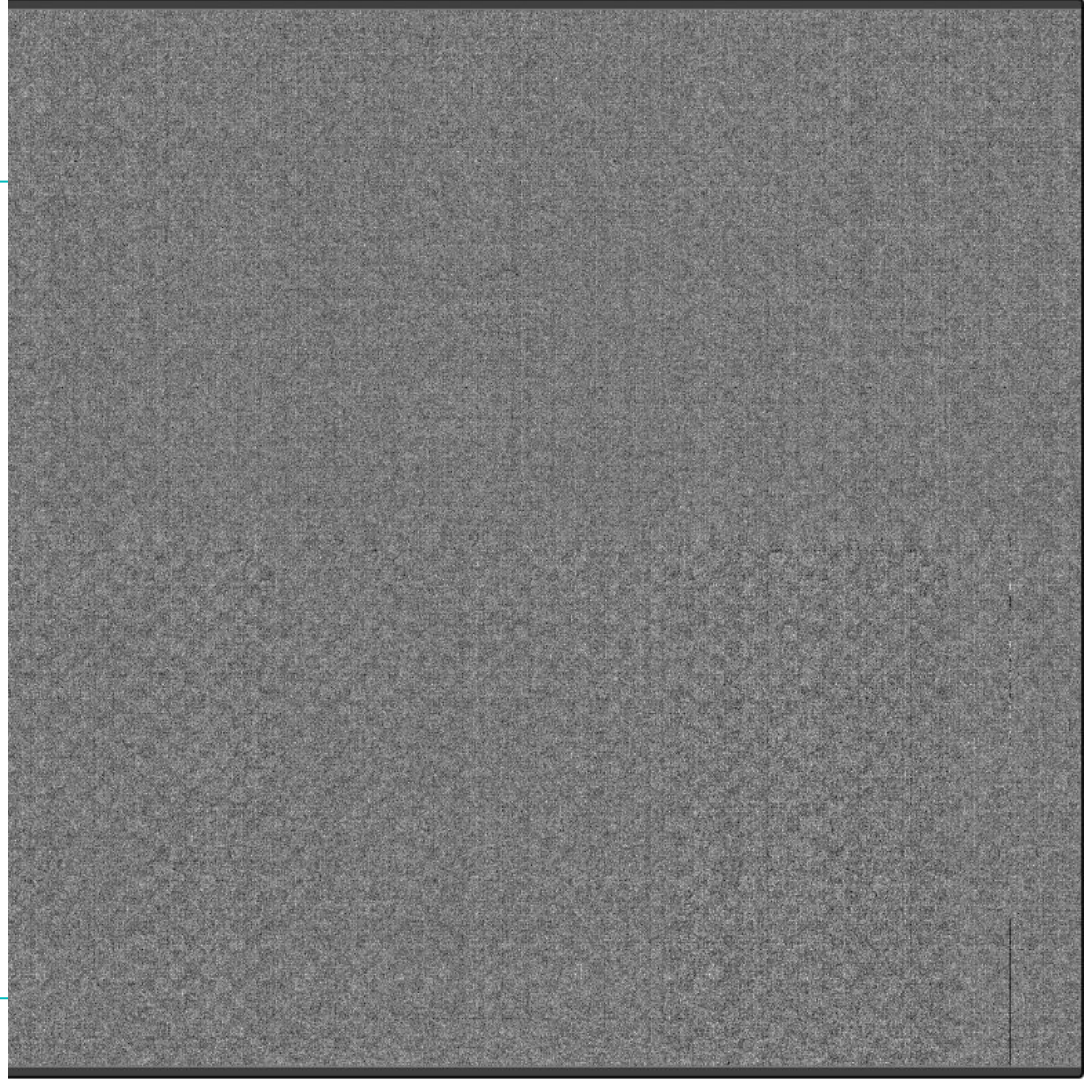
- First exposure stripes.



Examples:

Residual Bias:

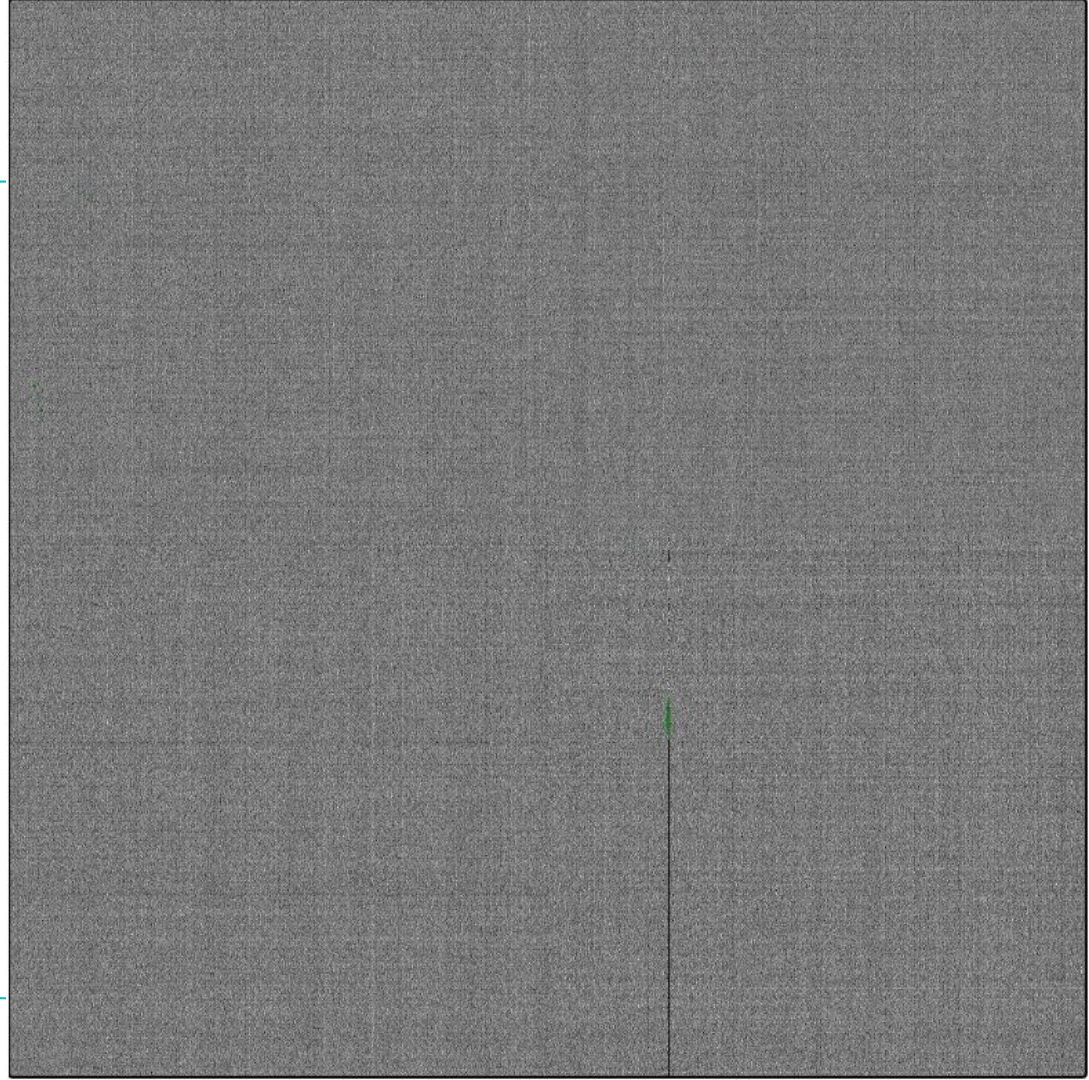
- Which are gone in the next exp.
- “Chessboard” isn’t real:



Examples:

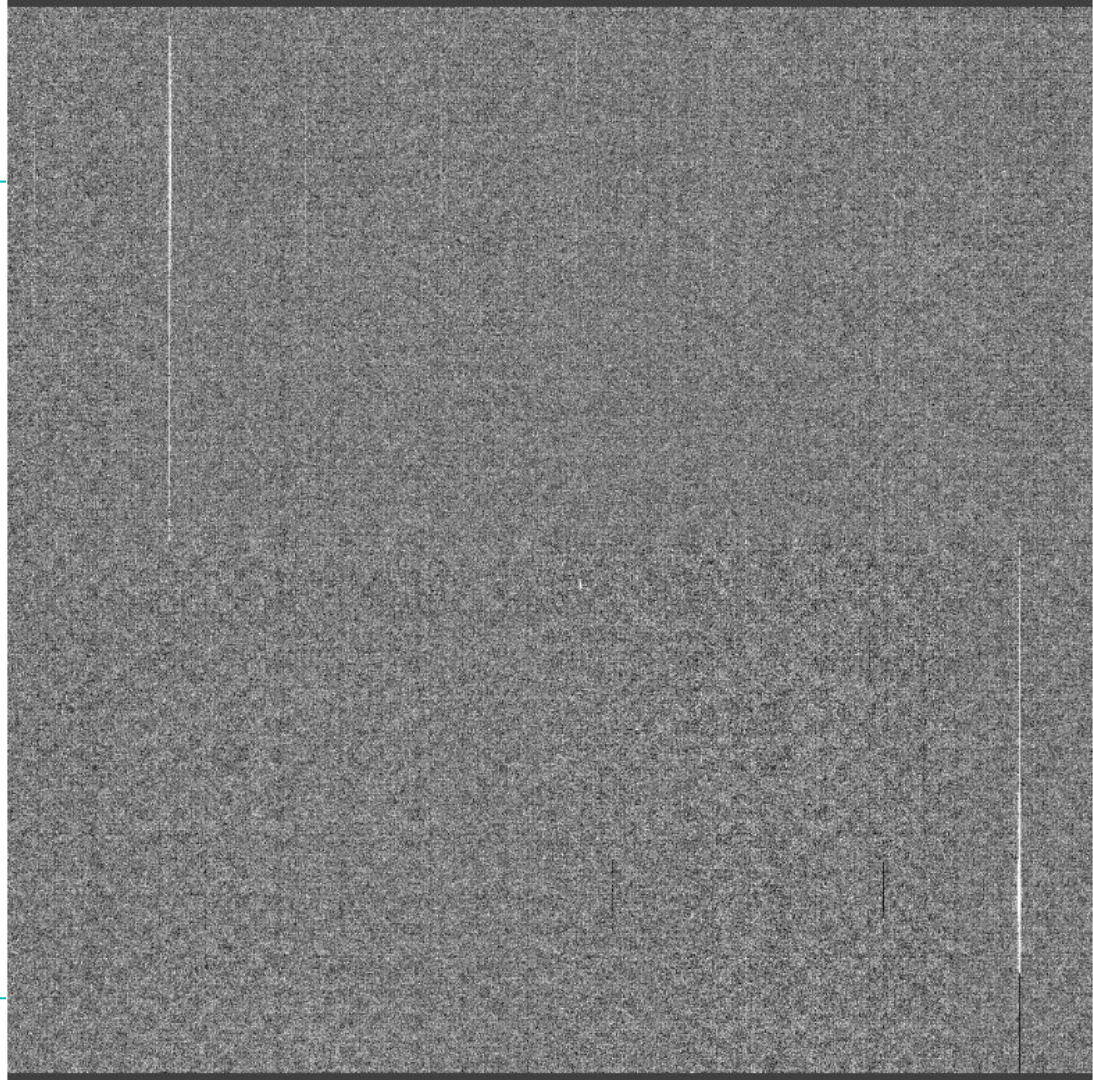
Residual Bias (full size):

- Which are gone in the next exp.
- “Chessboard” isn’t real:
  - It’s an artifact of the overscan



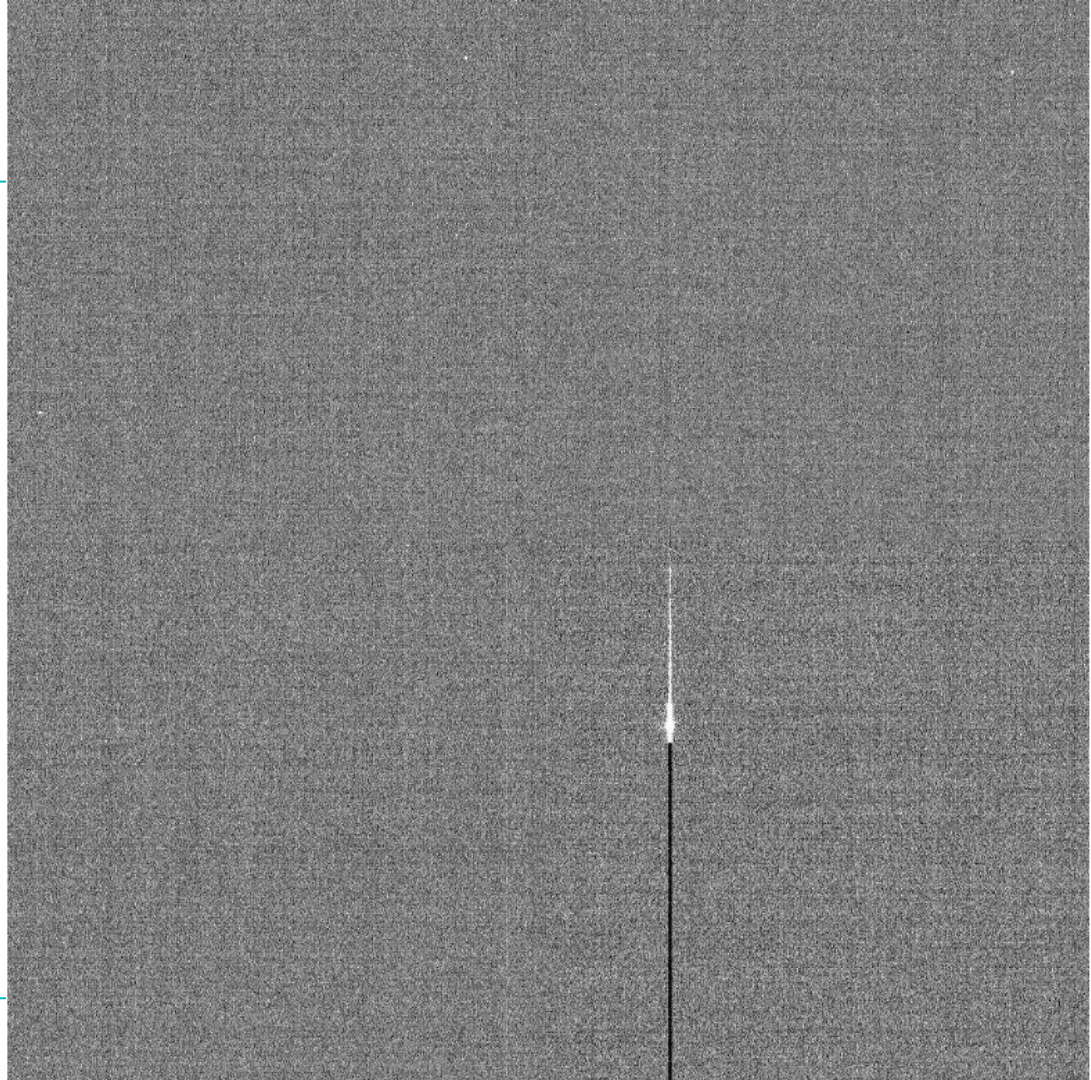
Examples:

Combined Dark



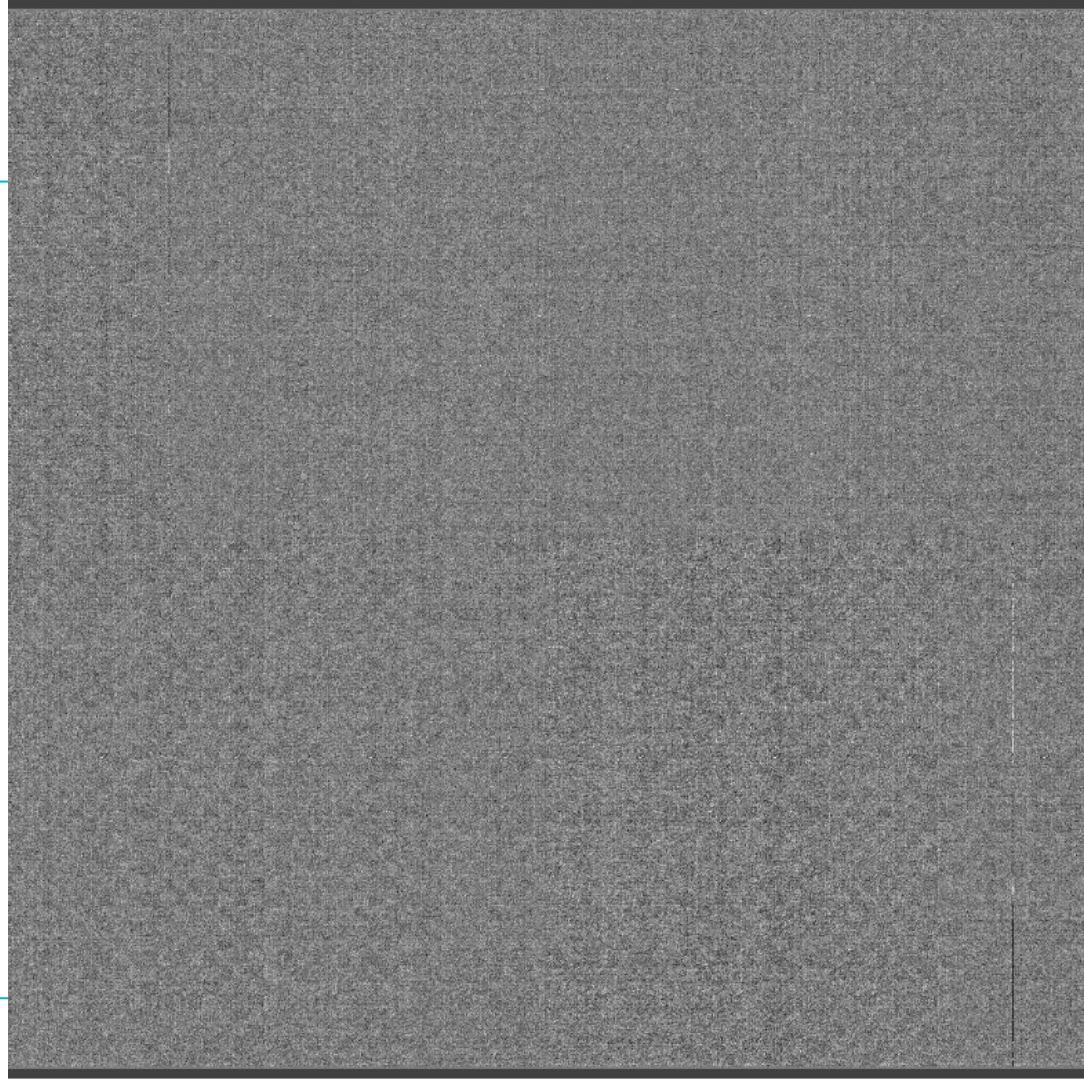
Examples:

Combined Dark (full size)



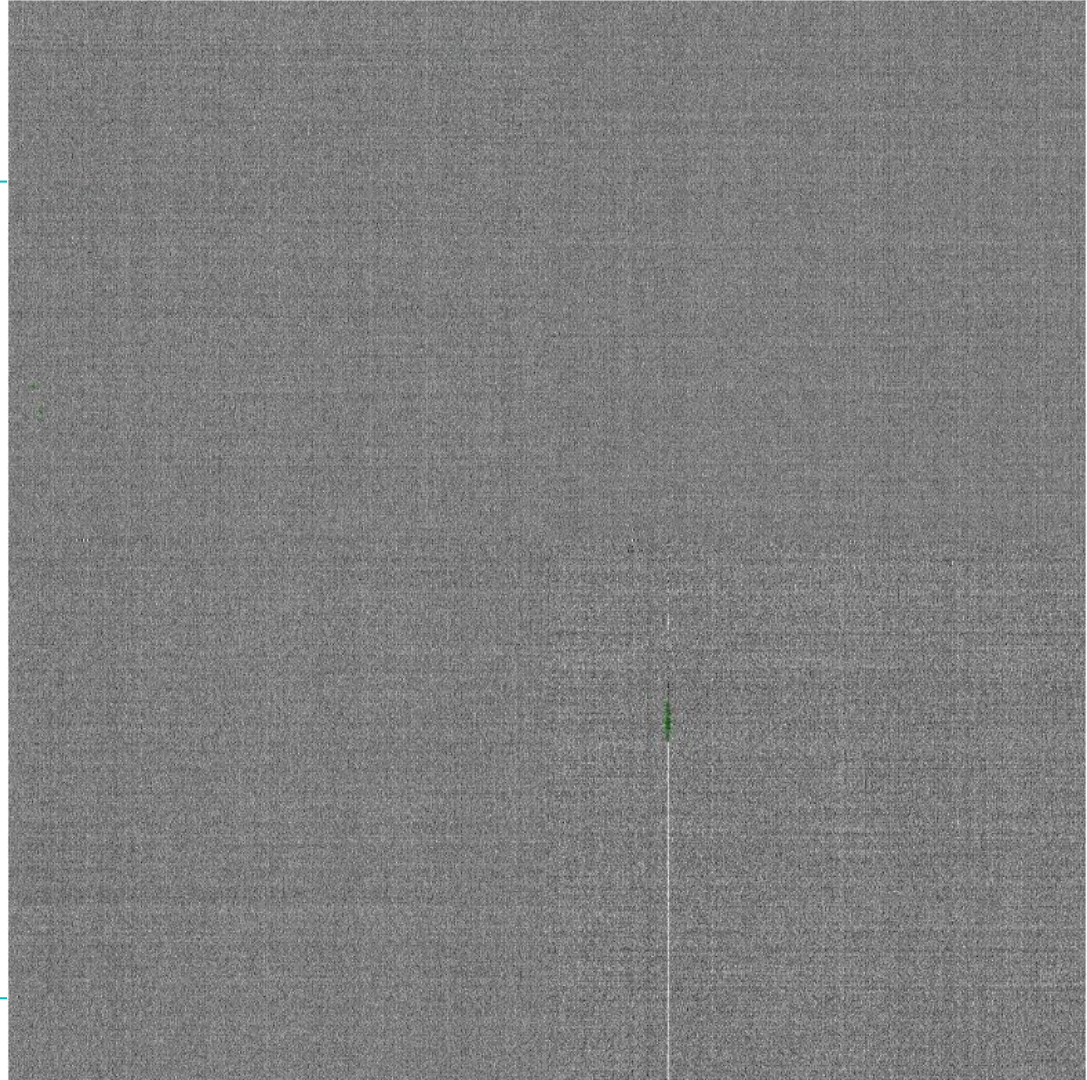
Examples:

Residual Dark



Examples:

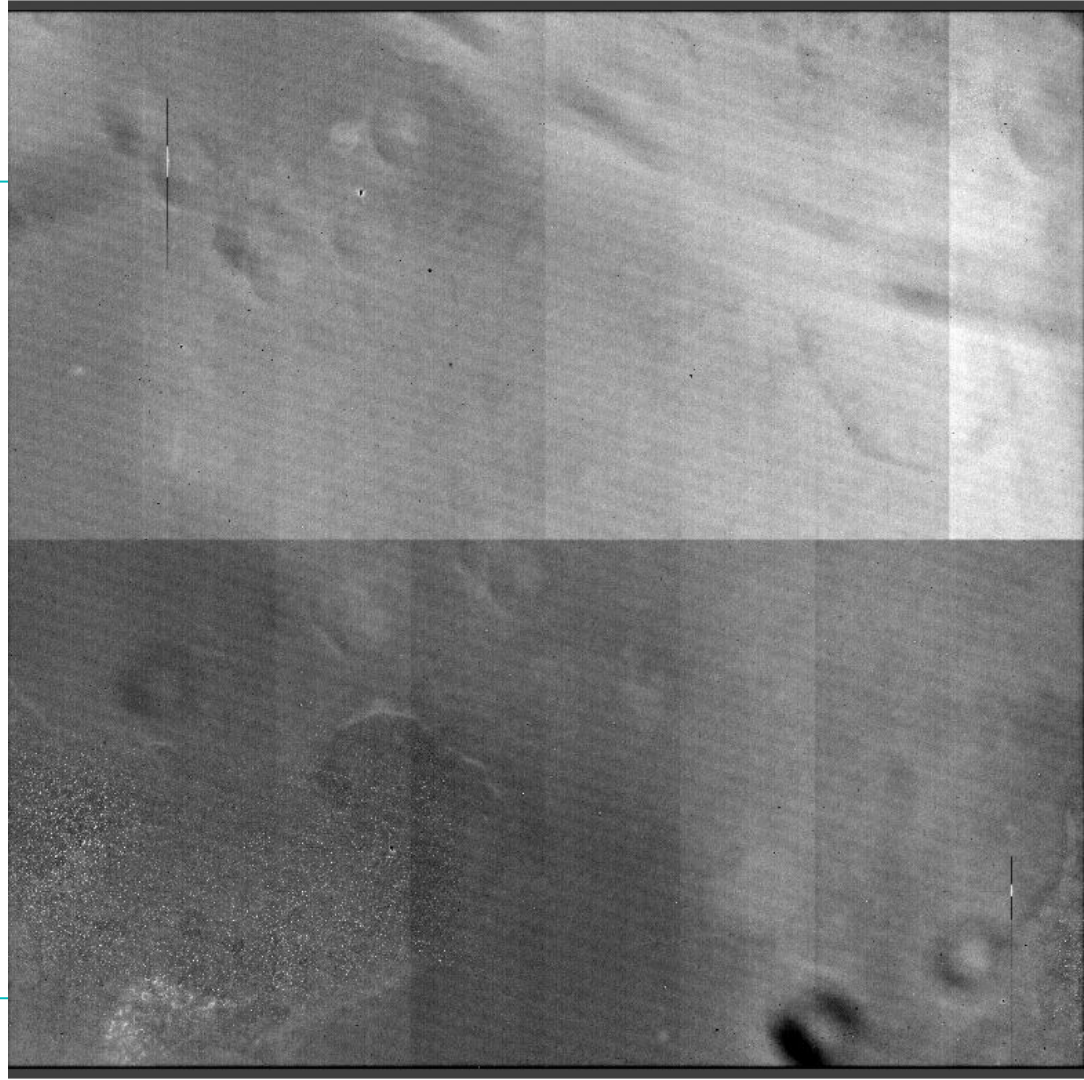
Residual Dark (full size)



Examples:

Combined flat (g)

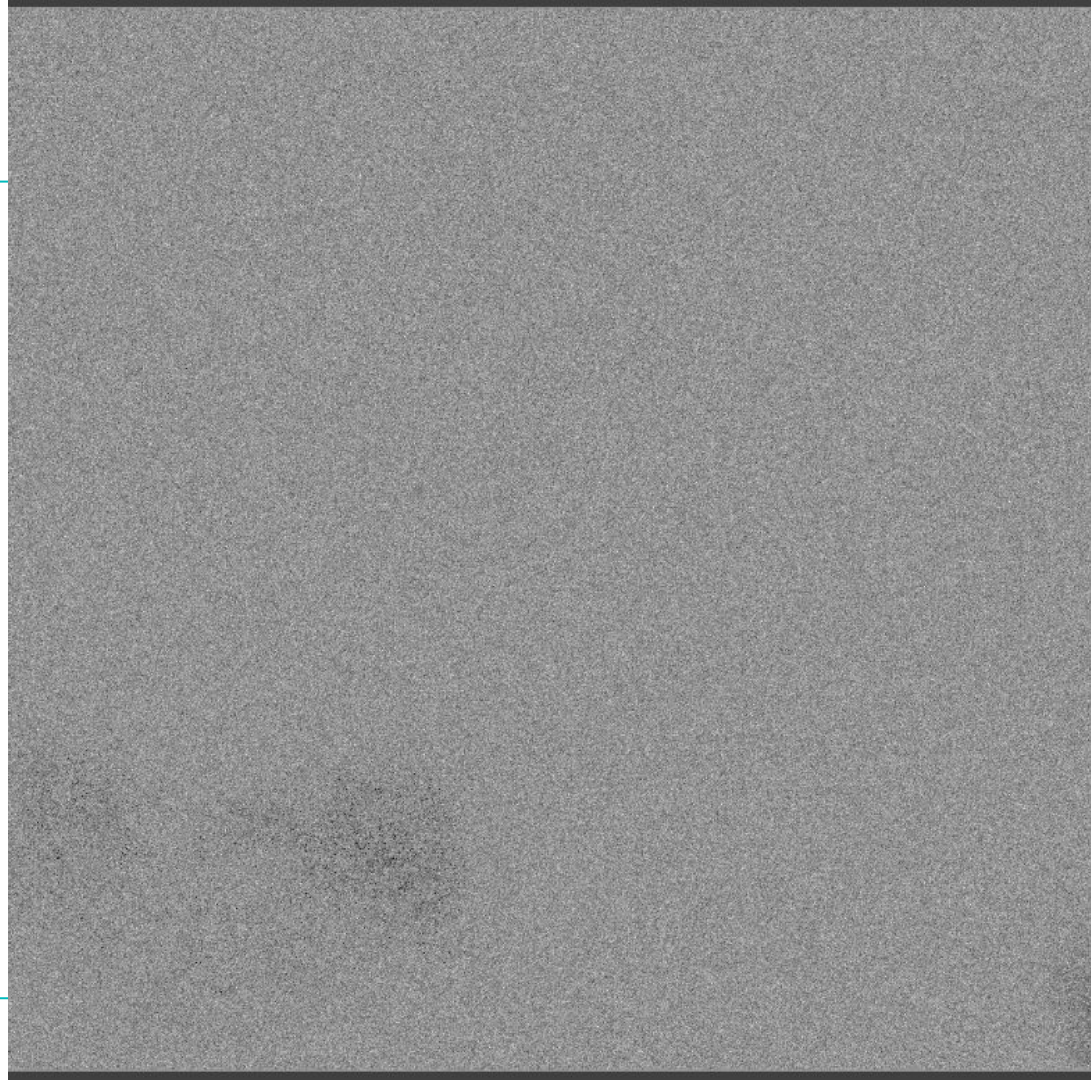
- Dust spots and such.



Examples:

Residual flat (g)

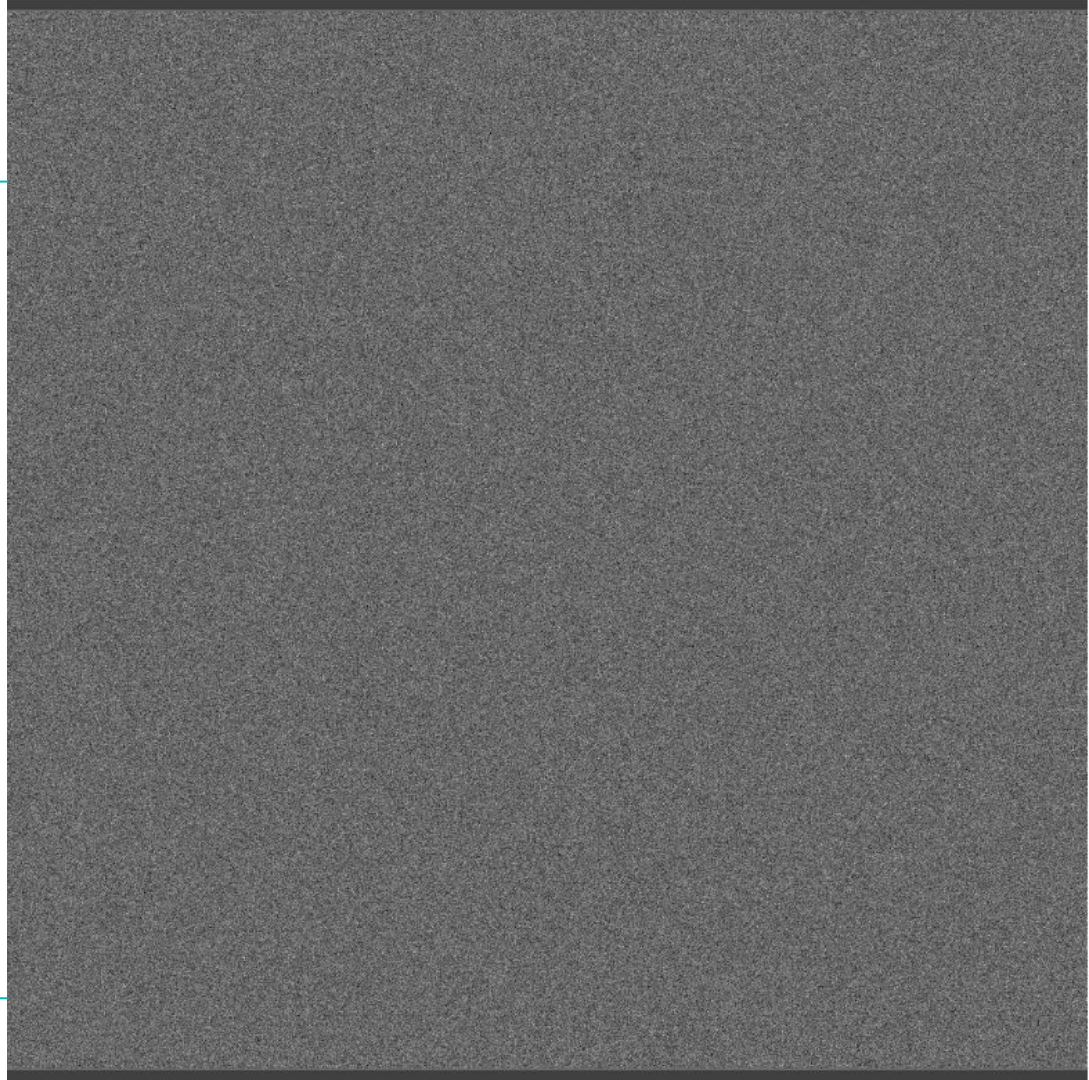
- First exposure.
- “Spicy corner”.
  - UV/blue end traps?



Examples:

Residual flat (g)

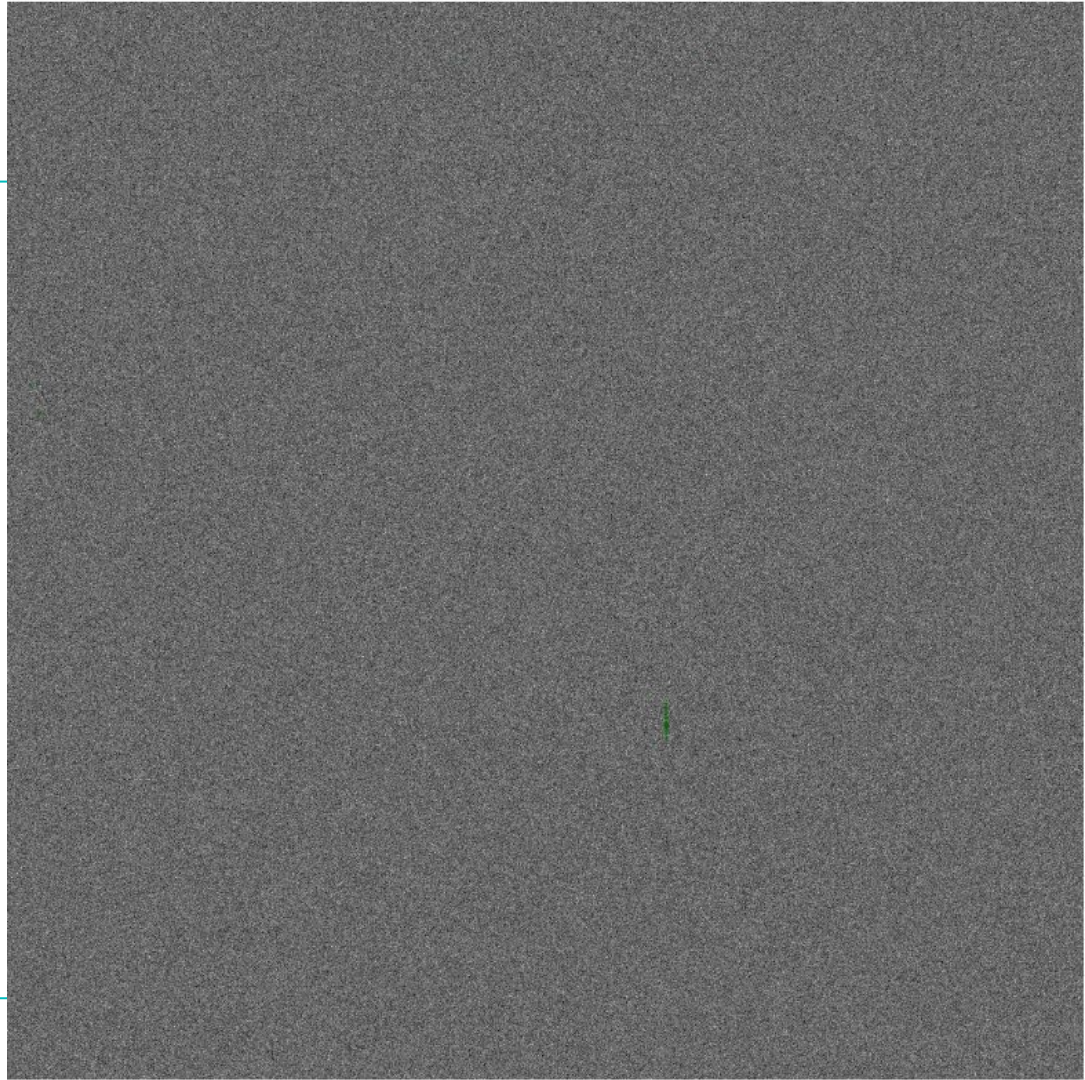
- First exposure.
- “Spicy corner”.
  - UV/blue end traps?
- Calms down on next exp.



Examples:

Residual flat (g; full size)

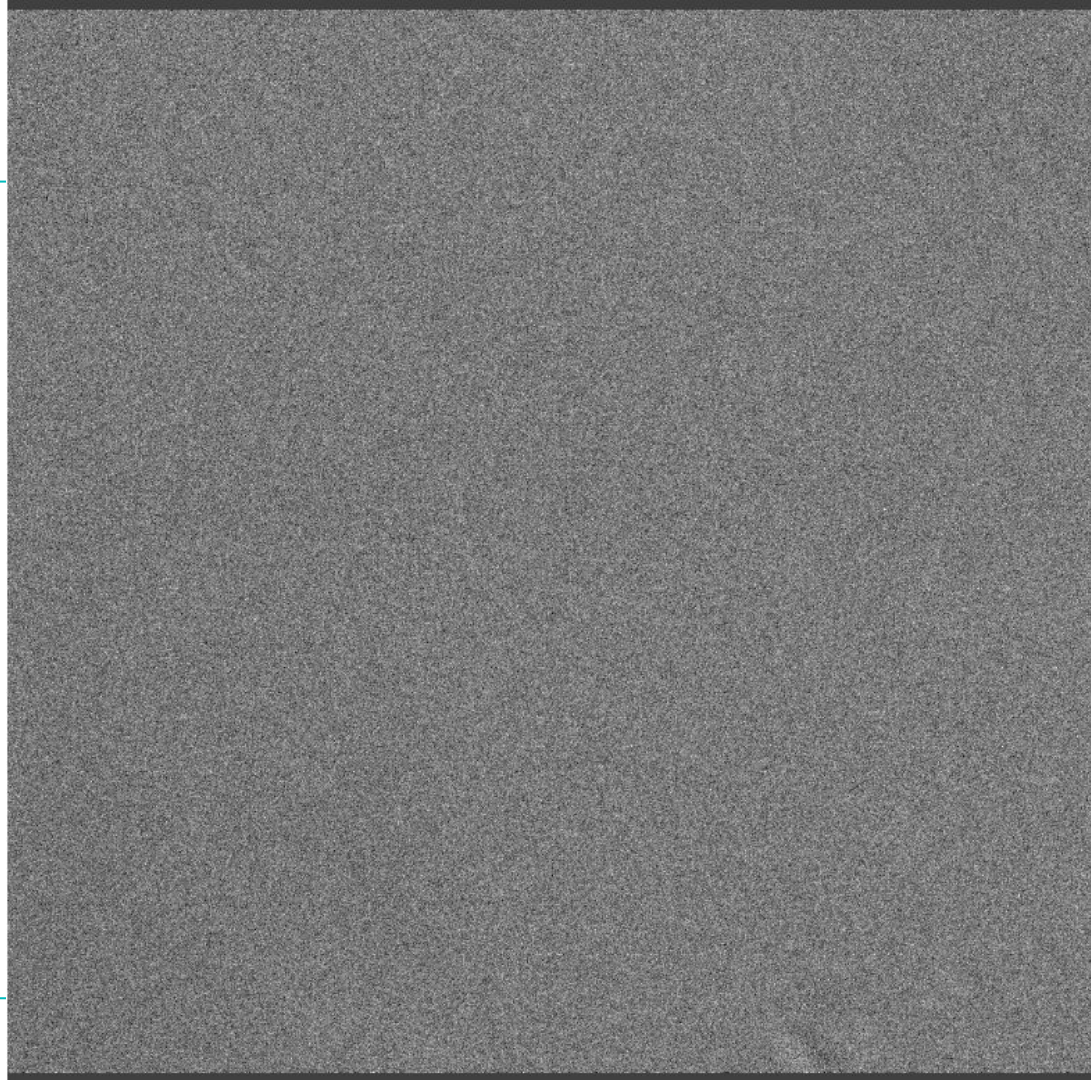
- First exposure.
- “Spicy corner”.
  - UV/blue end traps?
- Calms down on next exp.



Examples:

Residual flat (z; full size)

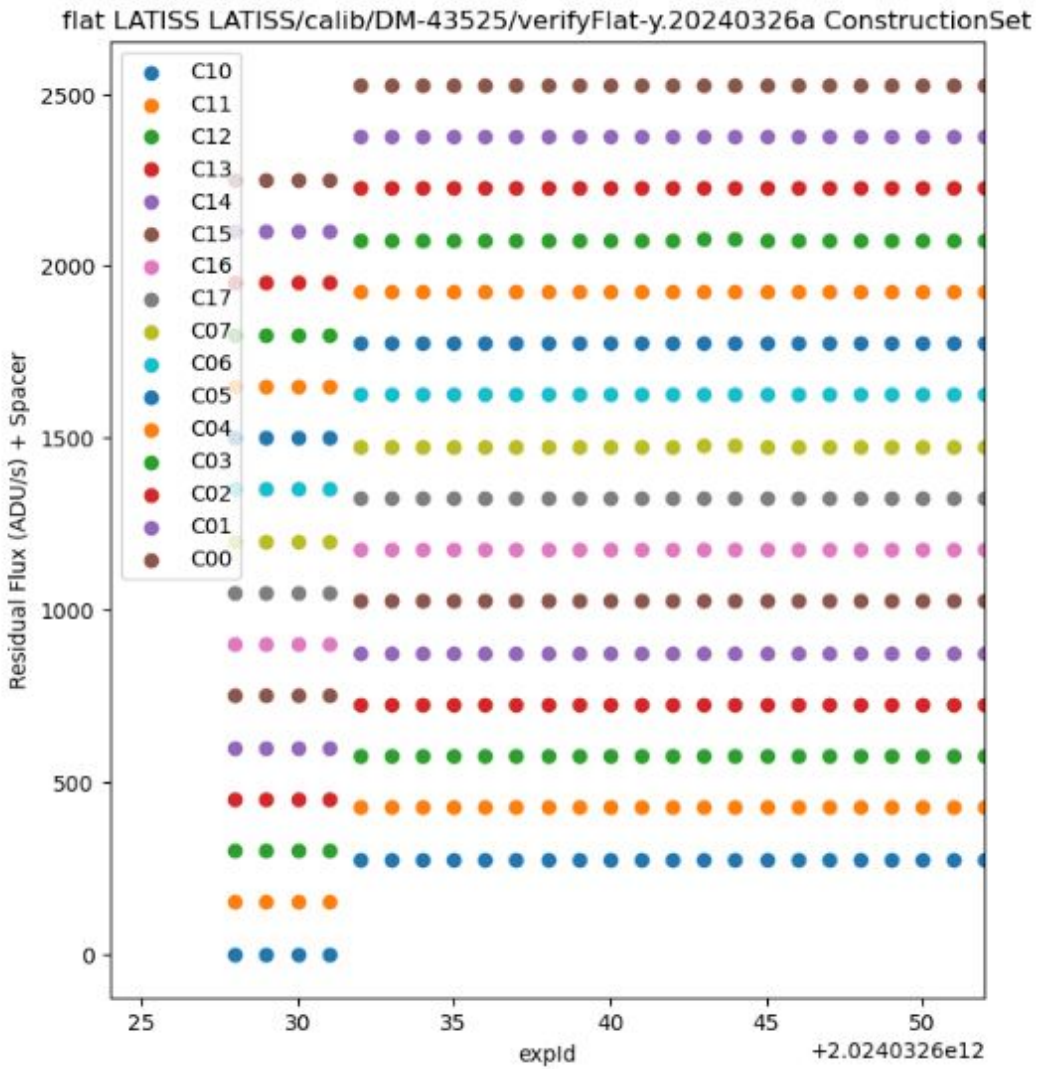
- Dust spot dipoles.



Examples:

Residual flat (y; mean(time))

- Why are those first four different?
- Because they're darks that were included by mistake.



End