Req 5.4.2

Title:

Master Domeflat

Objective:

Determine master dome flat frame for both **Keybands** and **Userbands**.

Master dome flats are obtained through an average with sigma rejection procedure on a stack of raw dome flats, intended to reduce photon shot noise, remove cosmic rays.

During the lifetime of **OmegaCAM** the dome flatfields shall be measured for the 4 keybands and the key-composite filter at least once/week. Thus at least within 3 days of the taking of science data a dome flatfield in the key passband will be available.

The sequence of master dome flatfields in the keybands, acquired over periods of months to years will be used to perform a trend analysis on the long term stability of the instrument and lamp which illuminates the dome flatfield. With the exception of the effects of the unstability of the lamp, this trend analysis is redundant with that obtained from both **req. 563** *Photometric Calibration - zeropoint key bands - doit* and that of **req. 562** *Photometric Calibration - monitoring*. Thus, by combining the results of these **req.**'s an accurate description of the behaviour of the lamp is feasible. The prediction of the behaviour of the lamp is a result, which will be used as an input for **req. 547** *Quick detector responsivity check*.

The dome flatfields will be used on an individual CCD chip level. The relative variations of the quantum efficiciency between individual CCD chips will be measured by **req. 563** *Photometric Calibration - zeropoint key bands - doit*.

The redundancy between various measurements of **req. 563** *Photometric Calibration - zeropoint key bands - doit* and **req. 542** *Master Domeflat* at keybands will be used to evaluate the relative chip-to-chip gain variations, and in due time, when advanced insight in this item is achieved, this knowledge might be used to further optimize observing scenarios.

The cross-calibration/handshaking of two lamp sets will be done during a transition period of about one week. During that week observations will be taken with both lamp sets. Template TSF–OCAM_img_cal_domeflat has a parameter which one of the two lamp sets will be used It is anticipated that about every 3 months a new set will become operational. During this transition

period of about a week both lamps sets will be observed sequentially. Also TSF-OCAM_img_cal_quick can be used for the crosscalibration.

For the userbands the dome flatfield will only be taken in the period of the week that that particular passband has been used for science observations. No trend analysis will be done on the data taken in the User pass bands.

It will be checked whether the dome flatfields are reproducable and whether artifacts appear on the same pixels. This will also serve as a check on the reproducibility of the filter exchange mechanism.

Fulfilling or fulfilled by:

Selfstanding.

Output is used by:
req. 563 *Photometric Calibration - zeropoint key bands - doit*
req. 547 *Quick detector responsivity* through the characterization of the dome flat field lamp.
req. 546 *Master flatfield*

When performed/frequency:

Daytime, daily. For the keybands the dome flatfields will be measured at least once/week. For the Userbands the dome flatfields will be measured at least within 3 days that the Userband has been used for science observations. When filters have been changed in the cassette the presence of dust and/or scratches might require new flat field exposures. During Comm. the reproducibility of the filter exchange mechanism will be checked.

Sources, observations, instrument configurations:

Domeflats with the four keyfilters and the single key composite filter. 5 observations per filter, with approximately 20000 ADU. Adding 5 exposures of 20,000 counts satisfies the accuracy requirement. Each science/standard observation preferably has an associated domeflat observed within 3 days. About 4 dome flats will be measured per day. The calibration lamps will be used and during a transition period of about one week both sets will be used.

Inputs:

Raw dome flatfields
CalFile– 541 *Master Bias frame*
CalFile– 522 *Hot pixel map*
CalFile– 523 *Gain*
CalFile– 535 *Cold pixel map*

Outputs:

CalFile– 542 *Master Domeflat frame*
CalFile– 542L *Dome Lamp*

Required accuracy, constraints:

Accurately measuring pixel-to-pixel gain variations as small as 1%. Re-insertion of the filter shall not alter the flat field stucture by more than 0.3% (rms, measured over the full detector area).

Estimated time needed:

Observation: 10 min. Reduction: 3 min./CCD.

Priority:

essential

TSF:

Mode– Stare N=5
TSF– OCAM_img_cal_domeflat

Recipe:

```
Dome_Flat -i <raw_domeflats> -b bias -g GAIN [-c cold] [-h hot]
          [-oc OVERSCAN_CORRECTION]

raw_domeflats       : raw dome flatfields
bias                : master bias frame
cold                : the cold pixel map
hot                 : the hot pixel map
GAIN                : the gain value in e/ADU (float).
                      Range of allowed values: 0.1 - 10
OVERSCAN_CORRECTION : overscan correction mode (integer).
                      Description of allowed values:
                        0: apply no overscan correction (default)
                        1: use median of the prescan in the
                           x-direction
                        2: use median of the overscan in the
                           x-direction
                        3: use median of the prescan in the
                           y-direction
                        4: use median of the overscan in the
                           y-direction
```

```
                        5: use the per-row value of the prescan
in

                           the x-direction

                        6: use the per-row value of the overscan
in

                           the x-direction
```

Before applying this recipe, use **Recipe– Split**—which is documented in **seq.–631**—with the `-t dome` option to split the raw multi-extension FITS input files.

Needed functionality:

image - processing (eclipse.trim_and_overscan)
image - arithmetic (eclipse.image_sub, eclipse.image_div, eclipse.image_pow, etc.)
image - cube median (eclipse.cube_median)
image - statistics (eclipse.stat)
image - mask (eclipse.image_threshold2pixelmap)
pixelmap - binary AND (eclipse.pixelmap_binary_AND)

CA:

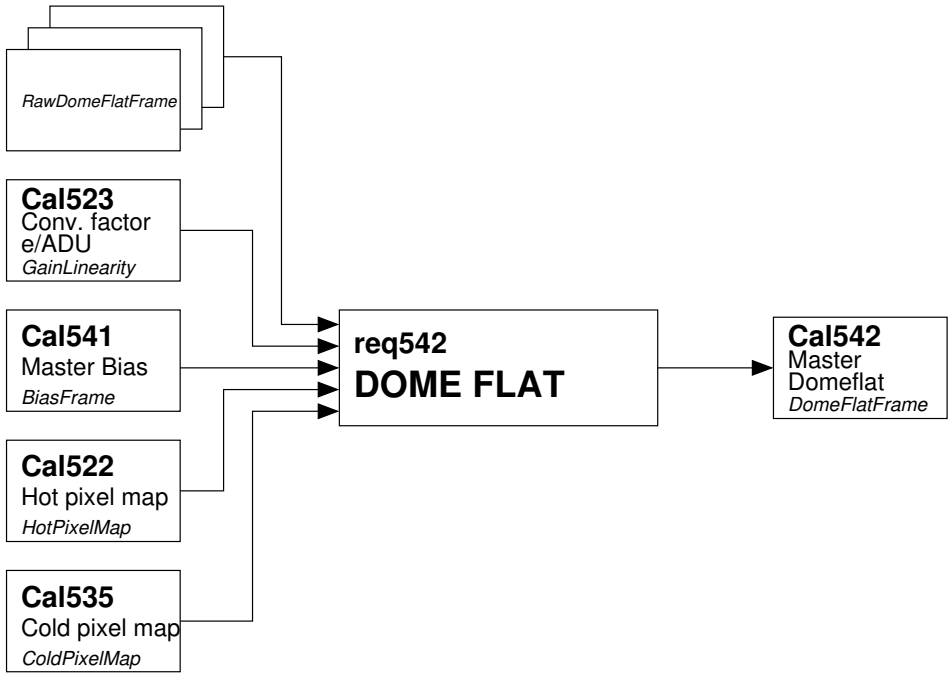

Fig 5.4.2 Dataflow and object class names (in small italic font) for req542

Process (make):

1. Construct a mask from input hot and cold pixel maps.
2. Trim, overscan correct and de-bias the raw dome frames.
3. Normalize frames to one illumination level (median pixel value 1.0).
4. Build a mean frame, rejecting 5σ outliers.

4.1 Compute the median frame of the stack of input frames.
4.2 Compute the error in each input frame ($\sigma = \sqrt{gain * counts}$)
4.3 Reject 5σ deviations in each input frame
4.3 Compute the mean in the remaining data

Verification (verify):
TBD

CAP:

```
Constants:
  SIGMA_CLIP      : the sigma clipping threshold

mask = eclipse.pixelmap_binary_AND(hot, cold)
```

```
scales = []
scaled_images = []

# Trim, overscan correct, de-bias and scale each input image
for raw_dome in raw_domeflats:
    reduced_dome = eclipse.trim_and_overscan(raw_dome)
    eclipse.image_sub_local(reduced_dome, bias)
    stat = eclipse.image_stat_opts(reduced_dome, pixelmap=mask)
    scaled_images.append(eclipse.image_mul(reduced_dome, 1.0/stat.median
    scales.append(1.0/stat.median)

median_image = eclipse.cube_median(scaled_images)

# Record the good values
good_data  = eclipse.make_image(xsize, ysize)
good_count = eclipse.make_image(xsize, ysize)

# Loop over remaining data
i=0
for dome in scaled_images:
    dev_image = eclipse.image_sub(median_image, dome)
    err_image = eclipse.image_pow(dome, 0.5))
    eclipse.image_div_local(dev_image, err_image)

    threshold = SIGMA_CLIP * sqrt(GAIN * scales[i])
    good = eclipse.threshold2pixelmap(dev_image, -threshold, thresh-
old)
    good = eclipse.pixelmap_2_image(good)

    eclipse.image_add_local(good_data,
                            eclipse.image_mul(dome, good))
    eclipse.image_add_local(good_count, good)
    i=i+1

#
domeflat = eclipse.image_div(good_data, good_count)
```