SUN/264.0

Starlink Project
Starlink User Note 264.0

Andrew G. Gibb & Tim Jenness

10 Jun 2014

ORAC-DR — SCUBA-2 Pipeline Data Reduction Version 1.4.0 User's Guide

Abstract

The ORAC-DR data reduction pipeline is designed to reduce data from many different instruments. This document describes how to use ORAC-DR to process data taken with the SCUBA-2 instrument on the James Clerk Maxwell Telescope.

Contents

1 Introduction

The SCUBA-2 pipeline is a suite of recipes and primitives for the ORAC-DR automated data processing software package. Additional functionality for processing and analysis of reduced data is provided through a number of PICARD recipes. General documentation on ORAC-DR and PICARD can be found in SUN/230 and **SUN/265** respectively.

The fundamental operation of the pipeline is to begin with raw data and produce calibrated science images with no additional user input. All decisions are based on metadata stored in the data files combined with basic quality assessment of reduced data products.

1.1 Document conventions

In an attempt to make this document clearer to read, different fonts are used for specific structures.

Observing modes are denoted by all upper case body text (e.g. FLATFIELD).

Starlink package names are shown in small caps (e.g. SMURF); individual task names are shown in sans-serif (e.g. makemap). ORAC-DR recipes and primitives are also shown in sans-serif and are always upper case (e.g. REDUCE_SCAN).

Content listings are shown in fixed-width type (sometimes called 'typewriter'). Extensions and components within NDF (SUN/33) data files are shown in upper case fixed-width type (e.g. `HISTORY`).

Text relating to filenames (including suffices for data products), key presses or entries typed at the command line are also denoted by fixed-width type (e.g. `% smurf`), as are parameters for tasks which are displayed in upper case (e.g. `METHOD`).

References to Starlink documents, i.e., Starlink User Notes (SUN), Starlink General documents (SG) and Starlink Cookbooks (SC), are given in the text using the document type and the corresponding number (e.g. SUN/95). Non-Starlink documents are cited in the text and listed in the bibliography.

File name suffices represent the text between the final underscore character and the three-letter `.sdf` extension. For example, a file named `s4a20101020_00002_0001_cal.sdf` has the suffix `_cal`.

2 SCUBA-2 Pipeline Variants

There are three variants of the SCUBA-2 pipeline, users will likly only need to run the science pipeline. The other two pipelines are designed to run in real time at the JCMT.

- The science pipeline has access to all the data observed for a given project and adopts a best-possible reduction approach. Images are made for each complete observation which are combined to create the final image.
- The quick-look (QL) pipeline is primarily designed to perform quality-assurance analysis of the incoming data for real-time assessment of the instrument performance. It is also responsible for processing pointing and focus data.
- The summit pipeline runs in parallel with the QL pipeline (though on a different machine) and is the primary image-processing pipeline. Processing is delayed until sufficient data exist to produce a higher quality image. In practice this happens after a certain time has elapsed since an image was last made.

2.1 Requirements for running the SCUBA-2 pipeline

The SCUBA-2 pipeline requires a recent Starlink installation. The latest version may be obtained from `http://starlink.eao.hawaii.edu/starlink`. Since development of the pipeline is an ongoing process, it is recommended that the newest builds be used to access the full capabilities of the pipeline. These builds can be obtained from `http://starlink.eao.hawaii.edu/starlink/rsyncStarlink` and may be kept up-to-date with rsync.

The Starlink Perl installation (Starperl) must be used to run the pipeline due to the module requirements. The Starlink environment should be initialized as usual before running ORAC-DR.

The software used to process raw data into images is called the SubMillimetre User Reduction Facility (SMURF). Detailed documentation on SMURF can be found in SUN/258, while SC/21 is a cookbook that describes some of the background to SCUBA-2 data reduction.

The pipeline uses the following Starlink applications:

- SMURF
- KAPPA
- FLUXES
- FIGARO
- CCDPACK
- CUPID

2.2 Important environment variables

The pipeline uses a number of environment variables to determine where data should be read from and written to. Some are set automatically when the pipeline is initialized, but they can be overridden manually and, with the `-honour` flag may be left unchanged between successive runs of the pipeline. The variables that must be defined in order for the pipeline to run are denoted as 'Mandatory' in the list below.

- `STARLINK_DIR`: location of the user's Starlink installation. [Mandatory]
- `ORAC_DATA_IN`: the location where data are read from. If running with `-loop flag`, this is the location of the flag files, rather than the data files. [Mandatory]
- `ORAC_DATA_OUT`: location where pipeline data products are written. Also used as a location for user-specified configuration files for makemap. [Mandatory]
- `ORAC_DATA_ROOT`: root location for data. At the JCMT, this is `/jcmtdata/raw/scuba2`. If not defined, the current directory is assumed. [Optional]
- `MAKEMAP_CONFIG_DIR`: a user-specified location for makemap configuration files. [Optional]
- `FINDCLUMPS_CONFIG_DIR`: a user-specified location for findclumps configuration files. [Optional]

As an example, to set up or override the pipeline environment variables, tcsh users will need to do:

```
% setenv ORAC_DATA_IN folder1
% setenv ORAC_DATA_OUT folder2
```

and bash users will need to do:

```
$ export ORAC_DATA_IN=folder1
$ export ORAC_DATA_OUT=folder2
```

2.3 Getting help

Basic help and a list of command-line options may be obtained after initializing ORAC-DR by running :

```
% oracdr -h
```

or

```
% oracdr -man
```

More complete documentation on ORAC-DR can be found in SUN/230.

3 Overview of the SCUBA-2 pipeline

3.1 Science pipeline

The science pipeline examines all the data files given to it and works out which files are related and should be processed together ("batch" mode). Each observation is still processed separately to produce an image, which is calibrated in mJy beam^{-1} (unless otherwise specified by the recipe). All the images for a given source are combined into a single coadd using inverse-variance weighting. If the source is a known calibrator then the images are checked for offsets and, if necessary, shifted to place the source at the correct position.

At the end of processing, all temporary files are deleted: the only data files left on disk will have the suffix `_reduced`.

Recipes exist for a number of different target types which contain different processing steps, relevant to each particular target type. These are bright or faint compact (such as planets, T-Tauri stars or extragalactic blank fields respectively) and extended (such as Galactic star-forming regions). Examples of these steps include applying a matched-filter to enhance point-source detectability or running a clump-finding algorithm.

Note that additional recipes exist for determining the noise properties of the science data. The first performs the same noise analysis and QA checks as the QL pipeline. This can be run offline by adding the recipe name ASSESS_DATA_NOISE to the ORAC-DR command line. The second recipe calculates the noise and NEP properties in addition to carrying out the standard map-making procedure, and compares the noise properties with that expected from the SCUBA-2 integration time calculator. This recipe is called REDUCE_SCAN_CHECKRMS. See the documentation below for further details.

3.2 Quicklook (QL) pipeline

The QL pipeline uses a task called `qlgather` to monitor the DRAMA processes and write out a flag file with the names of the files to process. The pipeline reads that flag file and processes each of the named files. `qlgather` collates data with the same subscan number. If new data are detected before all the expected files for the current subscan are in place, the new data take precedence and processing of the current subscan may be skipped.

Fastramp flatfields taken as part of each observation are processed and stored in the calibration system. The responsivity image for all four subarrays (along with the previous and percentage-change images) are displayed in a Kapview window.

Pointing and focus data are processed using the iterative map maker with a configuration file optimized for such observations of bright compact sources. Corresponding fastramp flatfields are obtained from the calibration system. For pointing observations an FCF is derived if the source is a known calibrator. The image is displayed using GAIA, showing in window 1, and then combined with the current coadd image (if one exists). The coadd image is displayed in another GAIA window (2), and its error component (noise) in another (3). For focus observations, the data for each SMU position is processed separately and combined into a three-dimensional data cube. The name of the pointing coadd or focus cube is written to a flag file (`.sYYYYMMDD_MMMMM.ok`, where `MMMMM` is the current observation number) for further analysis by the telescope POINTING_FOCUS task. The pipeline makes its own estimates of the pointing and focus solution, which are written to log files. Any temporary files created during this processing are deleted, keeping only files with a suffix `_reduced` or `_foc` for pointing and focus respectively.

Science data are processed as noise observations. The noise between 2 and 10 Hz is calculated along with the noise equivalent power (NEP) and the weighted NEP. These values undergo quality assurance checks to ascertain whether or not the instrument is still operating within specified limits. The focal-plane noise distribution is displayed in a Kapview window.

Data from other observing modes are processed as per their own recipes.

3.3 Summit pipeline

The summit pipeline is the main image-making pipeline running at the telescope. It will use all available data for processing (unlike the QL which will skip data to deal with the latest files), and as such may fall behind for several subscans. However, the processing is structured such that the processing for a given observation should be complete before a new observation begins.

Pointing and focus observations (along with setups, noise and skydips) are processed in a similar manner to the QL, though no data are skipped.

The pipeline checks for the presence of flag files which are updated with the names of new data as they are written to disk. As each file is picked up by the pipeline, it checks to see how much time has elapsed since the last map was made. If the time exceeds a threshold (typically about one-and-a-half to two minutes), then a new map is made using all of the data taken since the previous map. The map is calibrated in mJy beam^{-1} using the standard FCF. If it is too soon to make a new map, the raw data are flatfielded and left on disk (suffix `_ff`). These files will be deleted once a new image is made.

The new image is combined with the existing coadd (if appropriate) and a new NEFD image is calculated. Note that the summit pipeline coadds images for a given source across multiple observations so fainter features will be revealed as the integration time increases. If a new image was not created during the latest pass through the recipe then all flatfielded data files needed to create a map are left on disk.

4 Science pipeline processing of SCUBA-2 data

4.1 Running the science pipeline at your home institution

If the data for your project have been downloaded from CADC and placed in a single directory, the easiest procedure is to create a text file containing the name of each of these raw files. That file should contain either the full path to each file or relative to the current directory (or the directory defined by ORAC_DATA_IN). The data can be processed with the commands:

```
% oracdr_scuba2_XXX -cwd
% oracdr -files <list_of_files>
```

where XXX is the wavelength (450 or 850). An optional UT observation date in the form YYYYMMDD may be given (e.g. 20100301). If the date is omitted, the current date is assumed - however, the file naming convention uses the date on which the data were taken. The initialization command only needs to be run once per UT date, and may be given the -honour flag to use existing definitions of the relevant environment variables. Alternatively, the -cwd flag may be given to force the pipeline to use the current working directory for output.

Note that there is no need to uncompress the data files prior to running the pipeline: ORAC-DR can accept files compressed with gzip (ending .sdf.gz) and will uncompress them itself. However, be aware that the uncompressed files are not deleted at the end of processing (ORAC-DR does not delete raw data).

Each observation is processed separately and the images combined to form a single output image. If the list of files contains data from multiple sources, the pipeline will treat each source separately and create different output files accordingly. Calibration is handled automatically (see 4.3 below).

The default science recipes will display the individual observation images plus the final coadded image using GAIA. (The display can be turned off, if desired, by adding -nodisplay to the ORAC-DR command line.)

4.2 Pipeline products

The science data products from the pipeline have a suffix of _reduced. The files beginning with s are the products from individual observations; the files beginning with gs are the coadded observations for a single object. The products from non-science observations may have different suffices, and may be three-dimensional cubes. See the documentation on the individual recipes in Appendix F for further details on those products.

In addition to the data files, the reduced products have PNG format images 64, 256 and 1024 pixels on a side for easy viewing in an image viewer or web browser.

4.3 Calibration

If no calibration observations are available, and unless otherwise instructed, the pipeline will apply standard flux conversion factors (FCFs) to calibrate the images in mJy beam^{-1}. Currently these are 495000 mJy beam^{-1} at 850 μm and 472000 at 450 μm. (See also [3].)

4.4 Customizing the map-making

The pipeline uses the SMURF dynamic iterative map-maker (makemap) to create maps from raw data. A detailed description of the map-maker is given in SC/21 and [2]. The map-maker uses a configuration file to control the data processing which may be modified by users with advanced knowledge of the map maker. The SCUBA-2 pipeline may be given the name of an alternative or customized configuration file via the recipe parameter capability of ORAC-DR. A number of pre-defined configuration files exist in the directory `$STARLINK_DIR/share/smurf`.

Once a suitable configuration file has been created, add its name to a recipe parameter file as follows:

```
[REDUCE_SCAN]
MAKEMAP_CONFIG = myconfigfilename.lis
```

and add `-recpars recipe_params.lis` to the command line when running the pipeline, where `recipe_params.lis` is the name of recipe parameter file (which must be in the current directory if the path is not given). The makemap configuration file must exist in the current working directory or one of the directories defined by the environment variables `MAKEMAP_CONFIG_DIR`, `ORAC_DATA_OUT`, or `ORAC_DATA_CAL` or in `$STARLINK_DIR/share/smurf`. Each directory is searched in this order and the first match is used.

Note that if running a recipe other than REDUCE_SCAN (such as one of the dedicated JLS recipes) that recipe name should be placed in square brackets instead.

4.5 Running the science pipeline at EAO/JCMT

The raw data are stored at EAO in the same way as at the summit. (It is also possible to do this yourself at your home institution but in general will not be worth the effort: use the example above instead.) The machine `sc2dr5` is available at the summit for general user data processing.

If processing data from a single night, then ORAC-DR can be run with the `-loop flag` option to indicate that the pipeline should watch for the appearance of flag files (which end in `.ok`). The flag files contain the path to the files to be processed, and have a fixed naming convention so the pipeline can recognize them. Use the `-list` option to specify the observation numbers to be processed (otherwise the pipeline will begin at observation 1). The command `scuba2_index` will produce a summary of the available data.

If processing data from multiple nights, create a text file with the names of the relevant data files, as for running at your home institution above, and follow the same procedure.

4.6 Processing examples

To process a set of data downloaded from the JCMT archive at CADC, where the files to be processed have been listed in a text file called `mydata.lis`:

```
% oracdr -files mydata.lis
```

To process all files starting at observation 21 (skipping non-existent files) until there are no more files:

```
% oracdr -loop flag -from 21 -skip
```

To process the files from a list of observations (e.g. 21, 22, 28, 29 and 30):

```
% oracdr -list 21,22,28:30
```

Note the use of a colon to specify a contiguous range of observation numbers.

Two additional options are useful when running on a remote machine or when an X display is not available:

- Include the flag `-log sf` to print the pipeline output to a terminal rather than opening a separate window;
- Include the flag `-nodisplay` to disable the display of pipeline images.

5 Analysis of processed data

A variety of simple PICARD recipes (see **SUN/265**) exist to perform post-processing analysis.

SCUBA-2 specific PICARD recipes begin with or contain the word `SCUBA2`; recipes specific to processing JCMT data contain `JCMT`. Documentation for each recipe is given in **SUN/265**, and on the PICARD home page, `http://www.oracdr.org/oracdr/PICARD` where each recipe is fully documented.

Running PICARD is simple. For example:

```
% picard -recpars <recipe_param_file> RECIPE <list_of_files>
```

where `<recipe_param_file>` is a text file containing the relevant recipe parameters, `RECIPE` is the name of the recipe to run and `<list_of_files>` is the list of NDF files to process, which must exist in the current directory. The output files are written to the current directory, or the directory defined by `ORAC_DATA_OUT`.

Most recipes have one or more recipe parameters which can be specified using the `-recpars` option. Recipe parameters are given in a text file with the following format:

```
[RECIPE_NAME]
RECIPE_PARAMETER1 = value1
RECIPE_PARAMETER2 = value2
```

The available recipe parameters are listed in the documentation on the PICARD home page above and in **SUN/265**.

The recommended approach for a few common tasks is detailed below.

5.1 Coadding/mosaicking multiple images

Although the pipeline will mosaic observations of the same target from the same night, it is clearly desirable to combine data from multiple nights. Alternatively, the user may wish to exert some additional control over the mosaicking parameters.

The MOSAIC_JCMT_IMAGES recipe deals with processed JCMT data (including ACSIS data cubes) and takes into account the instrument-specific NDF components such as the exposure time (`EXP_TIME`). The choice of coadding task is left to the user and may be either CCDPACK makemos or KAPPA wcsmosaic (the default). If using makemos, images are first aligned using KAPPA wcsalign. By default, the images (and additional NDF components) are combined using a nearest-neighbour scheme but this may be overridden by specifying the relevant parameter for wcsmosaic or wcsalign.

The output mosaic takes its name from the last input file in the list and has the suffix `_mos`. The user should take care to ensure this file does not already exist otherwise it will be overwritten.

5.2 Registering images to a common centre

Random pointing offsets and drifts between observations on a given night (and over different nights) mean that the final mosaic of a point source will not be optimal, and any faint surrounding structure may be masked entirely.

The recipe SCUBA2_REGISTER_IMAGES is specific to SCUBA-2 data. The approach is to find the position of a given source in each image and apply a shift to the WCS so that the peak fitted positions are coincident for each image. If a suitable source exists in each image, this recipe should be used before mosaicking (above).

Several recipe parameters are required, namely the coordinates of the reference position. Currently only equatorial coordinates are supported and must be written in sexagesimal format. The registered images have the suffix `_reg`.

As with the mosaicking recipe, this recipe knows about and takes care of applying the same shift to the `EXP_TIME` and `WEIGHTS` (and `NEFD` if it exists) components, so the combined results are accurate.

5.3 Comparing noise with integration-time calculator

A PICARD recipe called SCUBA2_CHECK_RMS exists for making the same assessments as the ORAC-DR recipe REDUCE_SCAN_CHECKRMS. However it should be noted that this recipe should be run *only* on maps created from individual observations: it will not give the correct answer for coadds. This is because the coadds do not preserve the elapsed time, which makes it impossible to use the SCUBA-2 integration time calculator (ITC).

A minor difference is that, with the exception of running at EAO, it is not possible to determine NEP-based results. However, these are generally less useful for comparing with the ITC.

The recipe produces the same output log file, `log.checkrms` with the identical format. In order to calculate the NEFD, this recipe will also create an `NEFD` image within the given map unless one exists already.

6 Processing and analysis details

This section covers some aspects of the SCUBA-2 pipeline in detail.

6.1 Matched filter

The standard matched filter used by the SCUBA-2 pipeline is based on a compensated PSF or Mexican-Hat wavelet technique (e.g. [1]). The filter employs a two-component gaussian based on the telescope beam determined in [3] to determine the detection scale of the PSF. Both the map and PSF are smoothed using a single, larger gaussian to remove a local background, and the smoothed versions subtracted from each. The smoothing gaussian has a FWHM of $30''$ at $850\,\mu$m, $20''$ at $450\,\mu$m. (For the relevant PICARD recipes, the FWHM of the smoothing gaussian may be given as a recipe parameter.) The smoothed-and-subtracted input image is convolved with the identically processed PSF to produce the output image.

For those recipes that assess the point-source response as part of the processing (e.g. the jack-knife-based recipes), the matched filter will use a PSF derived from the images that include the artificial source. In these cases, the map (and PSF) will not be smoothed by a larger gaussian before the convolution.

6.2 NEFD image calculation

For image data, the pipeline calculates a corresponding image of the noise equivalent flux density (NEFD), defined as the square-root of the product of the exposure time and variance components. Thus each pixel, i, in the NEFD image is given by:

$$\mathrm{NEFD}_i = \sqrt{(t_{\mathrm{exp},i}\sigma_i^2)} \tag{1}$$

Since this image is calculated from components internal to the image, the NEFD image is written as an additional NDF component under the same extension as the exposure time and weights, i.e. `MORE.SMURF.NEFD`. Note that the calculation will overwrite any existing component of the same name.

6.3 Source-fitting

KAPPA beamfit is the main task used for fitting sources in order to calculate beam size, pointing offsets and flux conversion factors (FCFs). The facility exists (within PICARD) to attempt to fit a realistic beam using two (circular) gaussian components as determined in [3]. The criterion is that the peak signal-to-noise ratio (SNR) must exceed 100. See the documentation for SCUBA2_CHECK_CAL in **SUN/265** for further details.

When estimating the beam size, beamfit always assumes a gaussian profile whether or not it is fitting two components. Fits to the beam are always carried out in an Az-El coordinate frame so that fits may be analyzed for systematic elongations.

For calculating pointing offsets, the peak position is most important and the choice of profile has no effect on the result. FCF calculations will use a single-component fit and the profile is left as a free parameter.

6.4 FCF calculations

The pipeline calculates three FCFs to convert the uncalibrated data in pW to astronomically-meaningful units:

- ARCSEC – calibrate maps in surface brightness units, Jy arcsec^{-2};
- BEAM – calibrate maps in Jy beam^{-1};
- BEAMMATCH – calibrate maps processed with the matched filter in Jy.

All three of these FCFs are calculated in the `_FIND_CALIBRATION_MAP_` primitive with the detailed calculation of each carried out in the primitives specified below.

The combination of these FCFs can be used to assess telescope performance. The ratio of the BEAM FCF to the ARCSEC FCF provides an estimate of the effective solid angle of the telescope beam which can be compared with the standard value derived in [3]. If the telescope is well focussed, the two should agree to within the calibration uncertainty. However, if the focus is not optimal, the BEAM/ARCSEC ratio will yield a larger value.

Maps of calibrators are made with 1″ pixels at both 850- and 450 μm which allows the fitting areas to be defined in terms of pixels.

6.4.1 ARCSEC

The ARCSEC FCF is calculated using aperture photometry (autophotom) on a calibrator (using the `_APERTURE_PHOTOMETRY_` primitive). The primary aperture is 30″ in radius (at both wavelengths) with a sky annulus defined within 1.25–2.0 times the aperture radius. The known total flux of the source is divided by the measured background-corrected flux to yield the ARCSEC FCF in Jy arcsec^{-2} pW^{-1}.

The autophotom task is called with the following parameters:

```
biasle=0 centro padu=1 photon=3 positive skyest=2 nousemags nousemask
```

The input file defining the source position and aperture properties contains the following lines:

```
1 x y 0.0 0.0 0.0 0.0 OK r_ap 0.0 0.0 annulus circle
#ANN 1 1.25 2.0
```

where x and y are the RA and Dec of the source (obtained from the `skyref` WCS attribute) and $r_{\rm ap}$ is the radius of the aperture in pixels.

The signal sum, S, is obtained from the `SIGNAL` entry (column 7) in the output file, which is converted to a total flux (pW arcsec^2) using the pixel area, $F = SA_{\rm pix}$. The uncertainty in this flux is derived from the `MAG` and `MAGERR` entries (columns 4 and 5 respectively). With the `nousemags` parameter, these values are counts, rather than magnitudes and are thus a mean count (μ) and uncertainty in that value ($\delta\mu$). Then $\delta F = F\mu/\delta\mu$ (also in pW arcsec^2).

6.4.2 BEAM

The BEAM FCF is obtained from the ratio of the known peak flux to the fitted source peak to give the FCF in units of Jy beam^{-1} pW^{-1}. The fitted peak is derived from KAPPA beamfit called from the `_FIT_SOURCE_` primitive. If the source has a SNR exceeding 100 the map is fitted by two superposed gaussians to mimic the realistic telescope beam. The fallback position is that a single (non-gaussian) component is fitted if the SNR is less than 100.

The arguments to beamfit for a single component fit are:

```
gauss=false mode=interface variance=false fitarea=15 fixback=0
```

The `pos` parameter is set to either (0,0) for planets or the RA and Dec of the reference position for stationary sources. (For a two-component fit, the `pos2` parameter is the same as `pos`.)

The peak of the fit and its uncertainty are used to estimate the FCF and the corresponding uncertainty directly. beamfit also returns an estimate of the RMS deviation between the map and the fit; however, since the FCF is derived from the peak of the fit, the uncertainty in that value is preferred for estimating the uncertainty in the FCF (although in practice the two tend to be similar).

6.4.3 BEAMMATCH

The BEAMMATCH FCF is obtained from the ratio of the known total flux to the fitted source peak in an image processed by the matched filter, to give the FCF in units of Jy pW^{-1}. KAPPA beamfit is used to fit a single component, though the fit is not constrained to be a gaussian in order to estimate the peak as accurately as possible. For point sources it should yield the same value as the BEAM FCF. However, it is rarely used to calibrate data directly; the application of a matched filter is usually carried out on images calibrated with the BEAM FCF.

The beamfit arguments for deriving the BEAMMATCH FCF are:

```
gauss=false mode=interface variance=false fitarea=15 fixback=0
```

where `fitarea` is the smaller of 1.5$\times$FWHM or 15 pixels. A smaller fit area is used in order to limit the influence on the fit of the negative dip associated with the matched filter. The `pos` parameter is the same as that used for the BEAM FCF.

As with the BEAM FCF, the uncertainty in the peak of the fit is used to directly estimate the uncertainty in this FCF.

6.5 Error beam

The SCUBA-2 error beam is defined as the fraction of the total power that lies outside of an aperture defined by the FWHM, i.e.: $E = 1 - (S_{\mathrm{main}}/S_{\mathrm{total}})$. For the model JCMT beam, these values are 0.57 and 0.67 at 850- and 450 μm respectively.

The fluxes are calculated within apertures of radii equal to half the FWHM and the standard radius for calculating the ARCSEC FCF above (i.e. 30 arcsec). The annulus used for the background estimate is kept the same in both cases at 1.25 and 2.0 times the standard aperture radius.

References

[1] Cayón, L. et al., 2000, *Isotropic wavelets: a powerful tool to extract point sources from cosmic microwave background maps*, MNRAS, 315, 757 6.1

[2] Chapin E. L., et al., 2013, *SCUBA-2: iterative map-making with the Sub-Millimetre User Reduction Facility*, MNRAS, 430, 2545 4.4

[3] Mairs S. et al., 2021, *A Decade of SCUBA-2: A Comprehensive Guide to Calibrating 450um and 850um Continuum Data at the JCMT*, MNRAS, 430, 2534

4.3, 6.1, 6.3, 6.4

A Running the SCUBA-2 pipeline at the JCMT

At the summit the pipeline is normally started by the telescope support specialist (TSS) as normal user accounts do not have the access privileges to write to the data output directories.

There are four pipelines running at the telescope, a QL and summit version for each wavelength. Each pipeline runs on a separate data reduction (DR) machine (`sc2dr#` where # is 1–4). Raw data are stored in `/jcmtdata/raw/scuba2/sXX/YYYYMMDD`, where `sXX` is the subarray and `YYYYMMDD` is the current UT date. Reduced data are written to `/jcmtdata/reduced/dr#/scuba2_XXX/YYYYMMDD` where `dr#` is the number of the machine running the pipeline and `XXX` is either 850 or 450. The directory `/jac_sw/oracdr-locations` contains files that list the locations of the output directories for each pipeline (and therefore which DR machine is processing which pipeline). Note that the output directories are local to their host computers (though they are NFS-mounted by the other DR machines).

Each pipeline waits for new data to appear before processing, and processes all data automatically choosing the correct recipe based on the observation type (which may be modified by the particular pipeline being run).

A.1 Prerequisites

DRAMA must be running on the QL DR machines, and the DRAMA task names must be defined. The task names are communicated through the `ORAC_REMOTE_TASK` environment variable, which contains a comma-separated list of names. The usual form of an individual task name is `TASK@server`, e.g., `SC2DA8D@sc2da8d`. The task name is in upper case; the machine name serving the parameter in lower case.

A.2 Running the QL pipeline

The QL pipeline is started with the following commands (substitute 850 for 450 for the short wave pipeline in this and the summit pipeline examples):

```
% oracdr_scuba2_850_ql
% oracdr &
```

The QL pipeline is fed data via DRAMA parameters and must be told the names of the tasks to expect data from, as described above. QL-specific recipes will be used if present. A stripchart program, which plots a number of quantities derived by the QL pipeline as a function of time, is made available once the QL pipeline has been initialized. Type `xstripchart` to run. (Note that the stripchart is a separate task and is not part of the pipeline itself.)

A.3 Running the summit pipeline

The summit pipeline is started by:

```
% oracdr_scuba2_850_summit
% oracdr -loop flag -skip &
```

The summit pipeline reads the data files from flag files, and skips non-existent observations. Summit-specific recipes will be used if present. Should the pipeline need restarting, the `-from` argument must be given to tell the pipeline the observation number it should begin processing.

B Processing JCMT Legacy Survey data

Currently, three of the JCMT Legacy Surveys have recipes optimized for the goals of the surveys. Support for the others will be added in as timely manner as possible in response to survey input. PICARD recipes also exist which replicate the steps performed on the processed data.

B.1 Cosmology Legacy Survey (CLS)

The CLS recipe employs a "jack-knife" approach using independent halves of the data in order to estimate and remove residual noise on large spatial scales.

- Maps are made with a modified blank-field config file and coadded into a single map.
- An artificial gaussian source is inserted into the data and the maps are remade (and coadded to make a PSF map).
- The signal-only maps are divided into two groups that are coadded separately and subtracted to form a jack-knife map.
- The central portion of the jack-knife map is used to estimate the spatial power spectrum which is applied to the signal coadd to remove residual low-spatial frequency noise ("whitening").
- A matched filter is applied to highlight compact sources using a whitened version of the PSF map as the input PSF. A signal-to-noise ratio image is also calculated.

Currently, this recipe works best on single scanned fields (i.e. not a mosaic of multiple fields).

B.2 Survey Of Nearby Stars (SONS)

This recipe is very similar to that for CLS with an additional step at the beginning. This step makes maps for each 30-second subscan and calculates the noise level in those maps to determine the noisiest subscans which are ignored by the map-making step. (Note the option exists to use the time-series noise instead.)

For this recipe, it is recommended that the artificial source used to determine the FCF correction be offset from the centre to avoid contamination of the signal.

B.3 SCUBA-2 "Ambitious-Sky" Survey (SASSy)

Processing of SASSy data is focused on detecting compact sources with the aim of following up previously-unknown or interesting-looking detections with more sensitive observations to probe the detailed structure.

- Maps are made with a blank-field config file and coadded.
- A matched filter is applied to highlight compact sources (using the default PSF).
- A peak-detection task (findclumps) is run to identify 5-σ peaks, which are written to a catalogue file.

B.4 JCMT Plane Survey (JPS)

Data reduction is tailored to enable the recovery of bright emission on scales up to 480 arcseconds. Data are reduced using a modified bright extended config file and coadded.

C Processing non-science data

This section contains detailed information on how the SCUBA-2 pipeline processes non-science observations and can be ignored by users only interested in processing science data. Each of the observation types are discussed in turn with relevant details on how they are processed by the different forms of the pipeline.

C.1 FLATFIELD

Flatfield data may be taken in a standalone flatfield observation, but they are also taken as part of science observing, where a fast-ramp flatfield measurement is made at the beginning and end of the observation. A flatfield observations consists of a series of measurements of the bolometer signal in the presence of different input heater powers, either a set of discrete values or as a series of continuous ramps alternately increasing and decreasing the heater power in a sawtooth pattern.

Standalone flatfield observations are processed with the REDUCE_FLATFIELD recipe. A flatfield solution is derived for each subarray present in each observation and written to an NDF file with the suffix `_flat`. The pipeline reports the number of good bolometers and statistics of the responsivities, as well as how they have changed relative to the existing flatfield solution (i.e. that present in the raw data files). This information is also written to a log file called `log.flatfield`.

The new solution is also shown and compared with the previous one graphically. Figure 1 is an example of the display. The results for each subarray are combined into a focal-plane mosaic and shown alongside the mosaic for the current solution. Histograms of the responsivities for the proposed and current solutions are shown as images side-by-side on the same scale. In addition a percentage-change image is shown below (scaled between ±10 %), as well as histograms of the new and existing responsivities (again on the same scale).

Once complete, the pipeline writes a flag file in `ORAC_DATA_OUT` for the current observation containing the name of each successful flatfield solution. The flag file is a hidden file with the name `.sYYYYMMDD_MMMMM.ok` where `YYYYMMDD` is the UT date and `MMMMM` is the zero-padded observation number. This file is used by the telescope control system (TCS) at the JCMT to identify the new flatfields to use in subsequent observations.

C.1.1 Fast-ramps

The fast-ramp flatfields (FASTFLAT) taken as part of on-sky observing are included with the science data and processed as part of map-making, but may be processed separately using the REDUCE_FASTFLAT recipe to assess how much the responsivities change during an observation. The results are calculated and displayed as above, but in this case the second fast-ramp flatfield is compared with the first, and not with the internal flatfield solution.

Fast-ramp flatfields are also processed separately in the QL and summit pipelines and picked up as necessary for map-making or noise calculations.

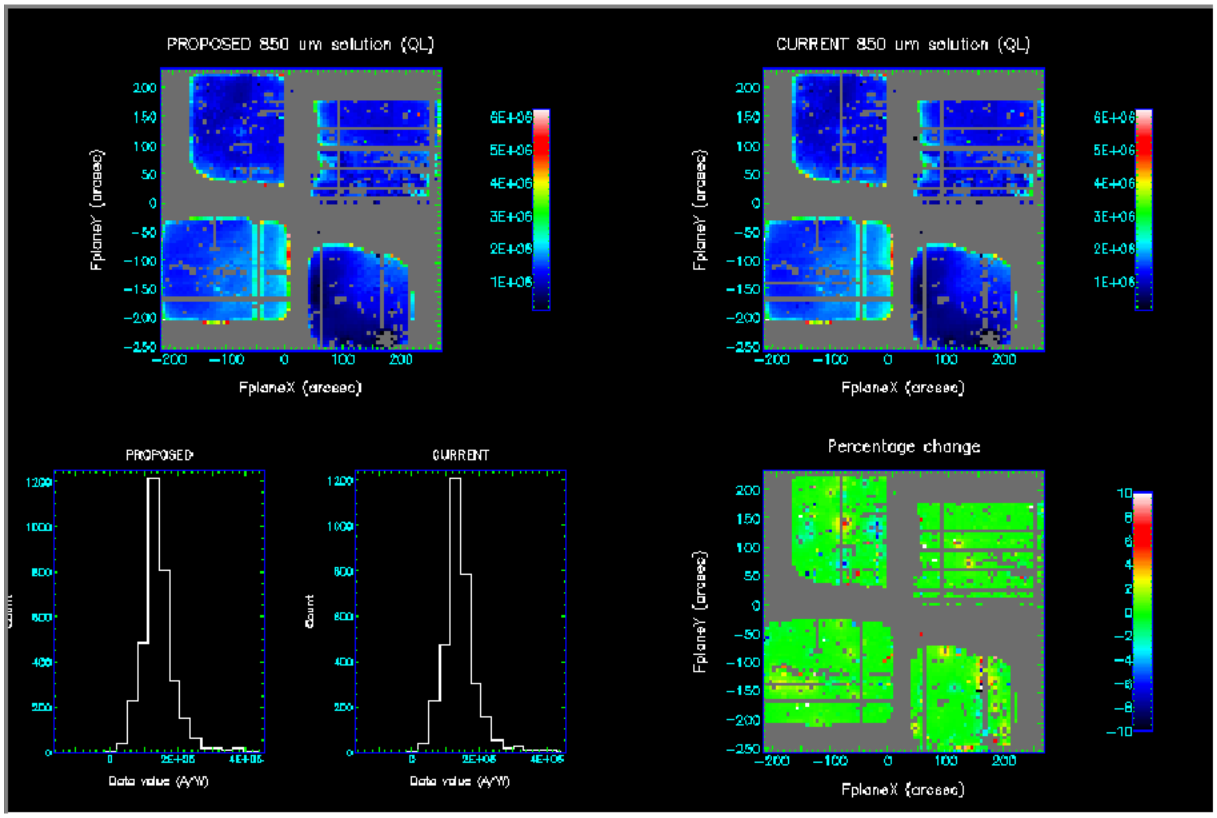

Figure 1: Example display from a FLATFIELD observation or FASTFLAT sequence. The top left panel is the responsivity map for this observation (containing the proposed new solution), the top right panel shows the current responsivity solution (i.e currently in use) for comparison. The bottom left panels are responsivity histograms for this observation and the current solution (PROPOSED and CURRENT respectively). The bottom right panel shows the percentage change in the responsivities between this observation and the current solution in use, scaled between $\pm 10\,\%$.

C.2 NOISE

Noise observations are a series of measurements recording the bolometer signal either against a dark shutter or open to the atmosphere. The telescope is not tracking a source during the measurement. The recipe calculates the average power spectrum of each array. The SMURF task calcnoise is used to calculate the noise properties for each subarray between 2 and 10 Hz. The recipe calibrates the noise data using the appropriate FCF to calculate the noise equivalent flux density (NEFD). If the measurement is on sky, the NEFD is also quoted for the zenith, using the current optical depth and airmass.

A numerical summary of the noise properties, showing the mean, median and modal noise values, the effective noise equivalent power and number of bolometers, is written to the screen. These results are also written to a log file called `log.bolonoise`. The NEFD results are written to a log file called `log.nefd`.

Focal-plane mosaics of the noise, the percentage change in the noise since the last measurement and the NEP are displayed in a Kapview window along with a histogram of the noise values (see Fig. 2).

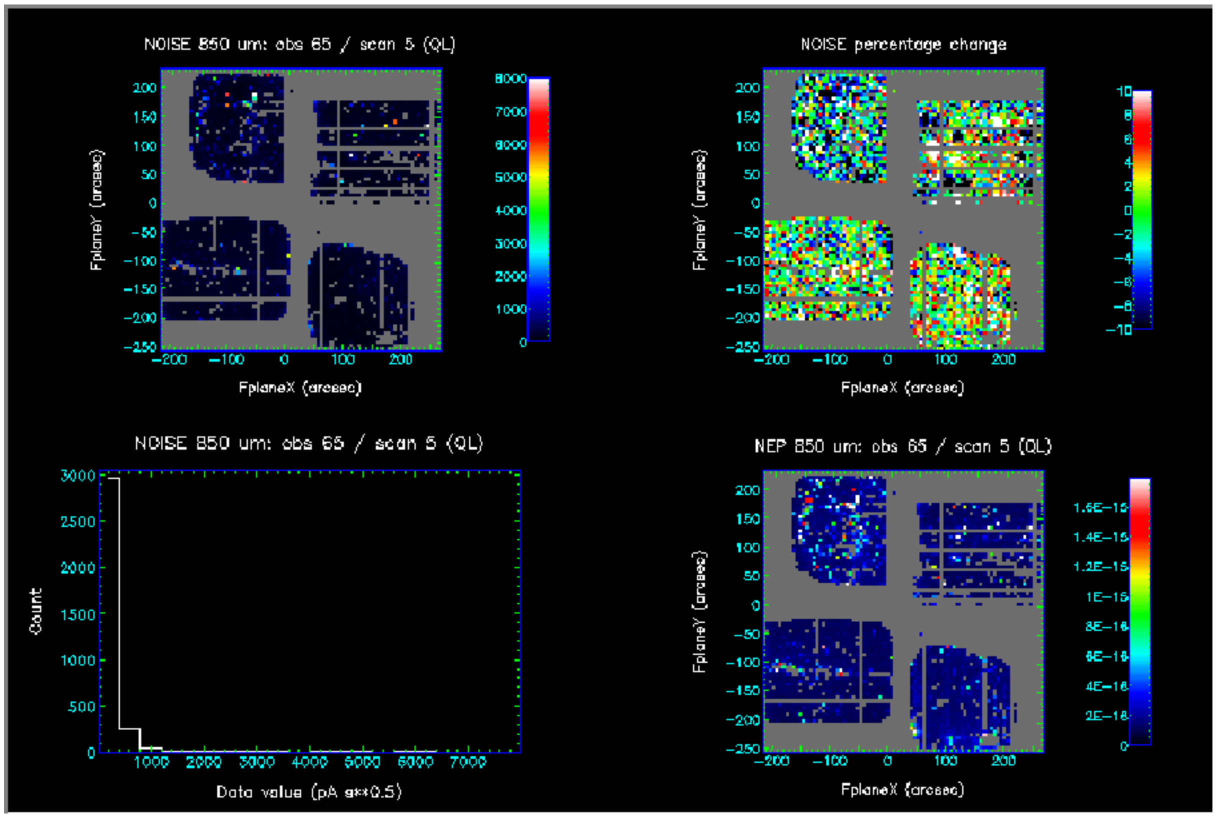

Figure 2: Example display from a NOISE observation. The top-left panel displays the noise for the current data, while the top-right panel shows the percentage change since the previous noise measurement. The bottom-left panel shows a histogram of noise values, while the bottom-right shows the NEP for the current data.

The QL and summit pipelines process each file as it is written to disk. Otherwise all of the data (for a given subarray) are passed to calcnoise.

C.3 POINTING

Pointing observations consist of making a map of a point source, and are processed with the REDUCE_POINTING recipe. The recipe processes the data with the iterative map-maker, crops the image to 75 arcsec on a side and removes any residual large-scale background signal with CUPID findback. If the source is a known calibrator, the pipeline derives an FCF.

The name of the processed image is written into a flag file (as for FLATFIELD observations above). The flag file is read by the telescope POINTING_FOCUS task which analyzes the named image to calculate the pointing offsets applied to the telescope pointing model.

The pipeline makes its own estimate of the pointing offsets in two ways by applying a point-source filter (matched filter) to enhance the signal-to-noise ratio in the image followed by fitting a 2-D profile to the source and calculating the centroid position. Both results are written to a log file called `log.pointing`.

The pipeline displays the image used to derive the pointing offsets in a GAIA window. The recipe also fits a 2-d profile to the source and writes the result to a log file called `log.beam`.

The map-maker uses the config file `dimmconfig_pointing.lis`, except for very short pointing observations (defined as $< 15\,\mathrm{s}$) which use `dimmconfig_veryshort_planet.lis` for planets and `dimmconfig_bright_compact_veryshort.lis` for all other sources.

The QL pipeline uses the REDUCE_POINTING_QL recipe, which contains different logic for dealing with DREAM and STARE data but is functionally identical to the main REDUCE_POINTING recipe for SCAN data.

C.4 FOCUS

Focus observations are processed with the REDUCE_FOCUS recipe. A focus observation consists of a series of maps made of a point source with the secondary mirror (SMU) at various positions. Only one axis (either X, Y or Z) is varied at a time.

In the QL pipeline, the maps are made as the data are taken and collated once the observation has ended. (In practice, due to the fact that the QL pipeline cannot rely on the `OBSEND` FITS header, an observation is considered to have ended once an image exists at each of the SMU positions: the number of SMU positions is retrieved from the FITS header.) The non-QL pipeline processes each SMU setting in turn.

Once all the maps exist, the pipeline creates a cube with the third axis given by the SMU position. The images are registered to the same pixel coordinates before adding to the cube so that they align correctly.

The pipeline writes a flag file (as above) which contains the name of the data cube. The flag file is read by the telescope POINTING_FOCUS task which analyzes the cube to calculate the best SMU position. The pipeline makes its own estimate of the best-fit SMU position by fitting a parabola to the peak fluxes derived from 2-D fits (provided at least three fluxes could be derived) and writes that number to a log file called `log.focus`.

The images at each focus position are displayed in sequence in a single Kapview window (see Figure 3).

The map-maker uses either the `dimmconfig_veryshort_planet.lis` for planets and `dimmconfig_bright_compact_veryshort.lis` for all other sources.

The same recipe is used for all forms of the pipeline. The output cube has the suffix `_foc`.

C.5 SETUP

A SETUP observation consists of a fast-ramp flatfield (FASTFLAT) measurement followed by a NOISE, both taken with the shutter closed. The recipe processes each separately as described for the respective observation types above. The results of processing each type of data are displayed in separate Kapview windows. The pipeline writes a flag file at the end of the observation with the names of the files containing the names of the flatfield solutions and noise calculations for the telescope software to read.

C.6 SKYDIP

A SKYDIP observation is a series of NOISE-SKY observations at differing airmass. Currently there is no pipeline recipe to process these data, other than as a NOISE observation.

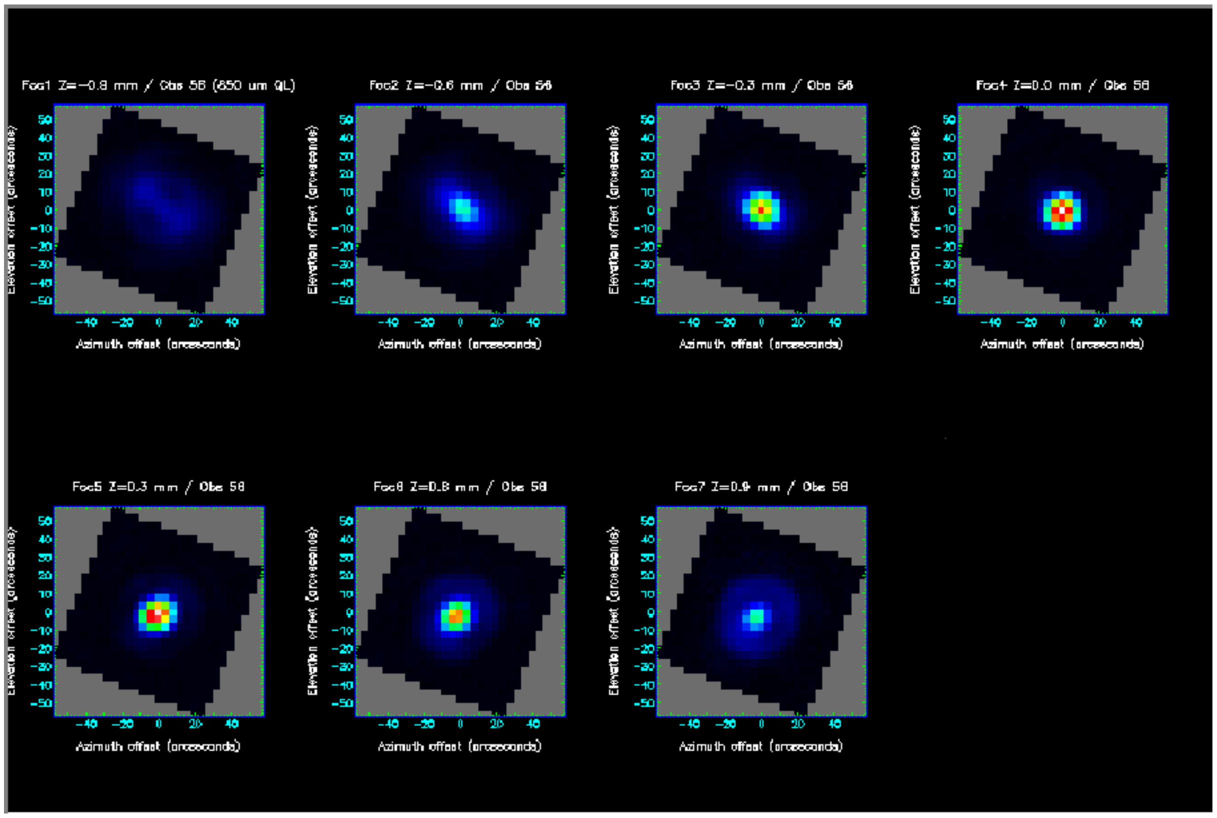

Figure 3: Example display from a FOCUS observation showing a reduced image for each of seven SMU positions. The SMU position (and axis) is given in the title for each image. The images are displayed with axes of arcsecond offsets.

D Engineering Data

Recipes also exist for processing engineering data, usually taken with the shutter closed and the telescope stationary. Historically, the real-time sequencer (RTS) was not involved in such measurements which prevented the pipeline from processing these data due to missing FITS headers and other essential metadata.

In September 2010 is was decided that the pipeline be used to process and analyze data taken in engineering mode, and a simulated RTS is used to generate metadata. Support for engineering data is currently quite rudimentary with only a single recipe available.

Note that in order to process data taken in engineering mode, the `-eng` option must be given to the ORAC-DR initialization script.

D.1 NEP

Measurements of the noise-equivalent power (NEP) are processed with the ENG_NEP recipe. The 'observation' consists of a series of NOISE measurements at different pixel heater and detector bias values. The recipe is designed to run offline, after the observation has finished.

The NEP is calculated for each subarray at each setting. These per-bolometer NEP images are combined to form a 4-d hypercube with axes bolometer row, bolometer column, heater setting and bias setting for each subarray. These hypercubes are named `sXXYYYYMMDD_MMMMM_nep.sdf` where `sXX` is the subarray label (e.g. `s8a`).

From these data, the effective and RMS NEP is calculated at each heater and bias setting, stored as a 2-d image with the heater and bias values as the X- and Y axes. These images have suffices of `_effnep` and `_rmsnep` respectively.

A number of log files are also written. The first is called `log.bolonoise` which contains the noise properties for each heater and bias setting and all subarrays. Log files for the 'mapping speed' parameter for each subarray are written separately with the name `log.mapspeed_SXX` where `SXX` is the subarray (e.g. s4a). The mapping speed is given by $N_{\mathrm{bol}}/\mathrm{NEP}^2_{\mathrm{RMS}}$ and is calculated for the best N_{bol} bolometers between 300 and 1000 in steps of 100 bolometers.

The recipe does not display any data.

E Alphabetical list of SCUBA-2 recipes

ARRAY_TESTS Co-add and display DREAM/STARE images

ASSESS_DATA_NOISE Calculate noise properties of SCUBA-2 data

BENCHMARK Benchmarking recipe

ENG_NEP Process NEP measurements made in engineering mode

ENG_NEP_QL Process NEP measurements made in engineering mode using the QL pipeline

FAINT_POINT_SOURCES Process SCAN data from faint compact sources

FAINT_POINT_SOURCES_JACKKNIFE Process blank field data with a jack-knife-based method

REDUCE_CLS Process data taken as part of the CLS JCMT legacy survey

REDUCE_DARK Process dark frames

REDUCE_DREAMSTARE Process DREAM/STARE images

REDUCE_DREAMSTARE_QL QL processing DREAM/STARE images

REDUCE_FASTFLAT Process fast-ramp flatfield data

REDUCE_FLATFIELD Process flatfield measurements

REDUCE_FOCUS Recipe for deriving focus information

REDUCE_FOCUS_QL process focus observations in the quick-look pipeline

REDUCE_FOCUS_SUMMIT derive focus information in the summit pipeline

REDUCE_FTS_FOCUS Recipe for processing FTS-2 focus observations

REDUCE_FTS_IMAGE_FOCUS Recipe for processing FTS-2 image focus observations

REDUCE_FTS_IMAGE_POINTING Recipe for processing FTS-2 image pointing observations

REDUCE_FTS_POINTING Recipe for processing FTS-2 pointing observations

REDUCE_FTS_SCAN Recipe for processing FTS-2 SCAN data

REDUCE_FTS_SCAN_QL Recipe for processing FTS-2 SCAN data in the QL pipeline

REDUCE_FTS_SCAN_SUMMIT Recipe for processing FTS-2 SCAN data in the SUMMIT pipeline

REDUCE_FTS_ZPD Recipe for processing FTS-2 SCAN data

REDUCE_FTS_ZPD_QL Recipe for processing FTS-2 SCAN data in the QL pipeline

REDUCE_FTS_ZPD_SUMMIT Recipe for processing FTS-2 SCAN data in the SUMMIT pipeline

REDUCE_JPS Process data taken as part of the JPS JCMT legacy survey

REDUCE_NOISE Process NOISE observations

REDUCE_NOISE_PHOTON Determine photon noise contribution to total noise

REDUCE_NOISE_QL QL processing NOISE observations

REDUCE_NOISE_SUMMIT SUMMIT processing NOISE observations

REDUCE_POINTING Process POINTING observations

REDUCE_POINTING_CADC Process pointing data at CADC

REDUCE_POINTING_QL QL processing of pointing observations

REDUCE_POINTING_SUMMIT Summit processing of pointing observations

REDUCE_POL_STARE Recipe for processing POL-2 stare data

REDUCE_POL_STARE_QL Recipe for processing POL-2 stare data in the QL pipeline

REDUCE_POL_STARE_SUMMIT Recipe for processing POL-2 stare data in the SUMMIT pipeline

REDUCE_SASSY Process data for the SASSy JCMT legacy survey

REDUCE_SASSY_QL Process data for the SASSy JCMT legacy survey in the QL pipeline

REDUCE_SASSY_SUMMIT Process data for the SASSy JCMT legacy survey in the SUMMIT pipeline

REDUCE_SCAN Process SCAN-mode data

REDUCE_SCAN_CHECKRMS Process scan data and collect noise/NEFD statistics

REDUCE_SCAN_EXTENDED_SOURCES Process SCAN data from extended sources

REDUCE_SCAN_EXTENDED_SOURCES_QL Process SCAN data from extended sources in the QL pipeline

REDUCE_SCAN_EXTENDED_SOURCES_SUMMIT Process SCAN data from extended sources in the SUMMIT pipeline

REDUCE_SCAN_FAINT_POINT_SOURCES Process SCAN data from faint compact sources

REDUCE_SCAN_FAINT_POINT_SOURCES_JACKKNIFE Process blank field data with a jack-knife-based method

REDUCE_SCAN_FAINT_POINT_SOURCES_QL Process SCAN data from faint compact sources in the QL pipeline

REDUCE_SCAN_FAINT_POINT_SOURCES_SUMMIT Process SCAN data from faint compact sources in the SUMMIT pipeline

REDUCE_SCAN_FAKEMAP Process SCAN data with existing map data added

REDUCE_SCAN_JSA_PUBLIC Form observation tiles for the public co-add

REDUCE_SCAN_QL QL process SCAN data

REDUCE_SCAN_SUMMIT Summit recipe for processing SCAN data

REDUCE_SETUP Process data from a setup observation

REDUCE_SETUP_QL Process data from a setup observation in the QL pipeline

REDUCE_SETUP_SUMMIT Process data from a setup observation in the SUMMIT pipeline

REDUCE_SKYDIP Process SKYDIP observations

REDUCE_SKYDIP_QL Process SKYDIP observations in the QL pipeline

REDUCE_SKYDIP_SUMMIT Process SKYDIP observations in the SUMMIT pipeline

REDUCE_SONS Process data taken as part of the SONS JCMT legacy survey

F Classified list of SCUBA-2 recipes

SCUBA-2 recipes may be classified in terms of their observing mode as follows.

Science data: SCAN mode

FAINT_POINT_SOURCES Process SCAN data from faint compact sources

FAINT_POINT_SOURCES_JACKKNIFE Process blank field data with a jack-knife-based method

REDUCE_CLS Process data taken as part of the CLS JCMT legacy survey

REDUCE_JPS Process data taken as part of the JPS JCMT legacy survey

REDUCE_SASSY Process data for the SASSy JCMT legacy survey

REDUCE_SASSY_QL Process data for the SASSy JCMT legacy survey in the QL pipeline

REDUCE_SASSY_SUMMIT Process data for the SASSy JCMT legacy survey in the SUMMIT pipeline

REDUCE_SCAN Process SCAN-mode data

REDUCE_SCAN_CHECKRMS Process scan data and collect noise/NEFD statistics

REDUCE_SCAN_EXTENDED_SOURCES Process SCAN data from extended sources

REDUCE_SCAN_EXTENDED_SOURCES_QL Process SCAN data from extended sources in the QL pipeline

REDUCE_SCAN_EXTENDED_SOURCES_SUMMIT Process SCAN data from extended sources in the SUMMIT pipeline

REDUCE_SCAN_FAINT_POINT_SOURCES Process SCAN data from faint compact sources

REDUCE_SCAN_FAINT_POINT_SOURCES_JACKKNIFE Process blank field data with a jack-knife-based method

REDUCE_SCAN_FAINT_POINT_SOURCES_QL Process SCAN data from faint compact sources in the QL pipeline

REDUCE_SCAN_FAINT_POINT_SOURCES_SUMMIT Process SCAN data from faint compact sources in the SUMMIT pipeline

REDUCE_SCAN_FAKEMAP Process SCAN data with existing map data added

REDUCE_SCAN_JSA_PUBLIC Form observation tiles for the public co-add

REDUCE_SCAN_QL QL process SCAN data

REDUCE_SCAN_SUMMIT Summit recipe for processing SCAN data

REDUCE_SONS Process data taken as part of the SONS JCMT legacy survey

Science data: DREAM/STARE mode

REDUCE_DREAMSTARE Process DREAM/STARE images

REDUCE_DREAMSTARE_QL QL processing DREAM/STARE images

Science data: FTS-2

REDUCE_FTS_SCAN Recipe for processing FTS-2 SCAN data

REDUCE_FTS_SCAN_QL Recipe for processing FTS-2 SCAN data in the QL pipeline

REDUCE_FTS_SCAN_SUMMIT Recipe for processing FTS-2 SCAN data in the SUMMIT pipeline

REDUCE_FTS_ZPD Recipe for processing FTS-2 SCAN data

REDUCE_FTS_ZPD_QL Recipe for processing FTS-2 SCAN data in the QL pipeline

REDUCE_FTS_ZPD_SUMMIT Recipe for processing FTS-2 SCAN data in the SUMMIT pipeline

Science data: POL-2

REDUCE_POL_STARE Recipe for processing POL-2 stare data

REDUCE_POL_STARE_QL Recipe for processing POL-2 stare data in the QL pipeline

REDUCE_POL_STARE_SUMMIT Recipe for processing POL-2 stare data in the SUMMIT pipeline

Pointing observations

REDUCE_FTS_IMAGE_POINTING Recipe for processing FTS-2 image pointing observations

REDUCE_FTS_POINTING Recipe for processing FTS-2 pointing observations

REDUCE_POINTING Process POINTING observations

REDUCE_POINTING_QL QL processing of pointing observations

REDUCE_POINTING_SUMMIT Summit processing of pointing observations

Focus observations

REDUCE_FTS_FOCUS Recipe for processing FTS-2 focus observations

REDUCE_FTS_IMAGE_FOCUS Recipe for processing FTS-2 image focus observations

REDUCE_FOCUS Recipe for deriving focus information

REDUCE_FOCUS_QL process focus observations in the quick-look pipeline

REDUCE_FOCUS_SUMMIT derive focus information in the summit pipeline

Noise observations

ASSESS_DATA_NOISE Calculate noise properties of SCUBA-2 data

ENG_NEP Process NEP measurements made in engineering mode

ENG_NEP_QL Process NEP measurements made in engineering mode using the QL pipeline

REDUCE_NOISE Process NOISE observations

REDUCE_NOISE_PHOTON Determine photon noise contribution to total noise

REDUCE_NOISE_QL QL processing NOISE observations

REDUCE_NOISE_SUMMIT SUMMIT processing NOISE observations

REDUCE_SKYDIP Process SKYDIP observations

REDUCE_SKYDIP_QL Process SKYDIP observations in the QL pipeline

REDUCE_SKYDIP_SUMMIT Process SKYDIP observations in the SUMMIT pipeline

Flatfield observations

REDUCE_FASTFLAT Process fast-ramp flatfield data

REDUCE_FLATFIELD Process flatfield measurements

Setup observations

REDUCE_SETUP Process data from a setup observation

REDUCE_SETUP_QL Process data from a setup observation in the QL pipeline

REDUCE_SETUP_SUMMIT Process data from a setup observation in the SUMMIT pipeline

QL-pipeline recipes

ENG_NEP_QL Process NEP measurements made in engineering mode using the QL pipeline

REDUCE_DREAMSTARE_QL QL processing DREAM/STARE images

REDUCE_FOCUS_QL process focus observations in the quick-look pipeline

REDUCE_FTS_SCAN_QL Recipe for processing FTS-2 SCAN data in the QL pipeline

REDUCE_FTS_ZPD_QL Recipe for processing FTS-2 SCAN data in the QL pipeline

REDUCE_NOISE_QL QL processing NOISE observations

REDUCE_POINTING_QL QL processing of pointing observations

REDUCE_POL_STARE_QL Recipe for processing POL-2 stare data in the QL pipeline

REDUCE_SASSY_QL Process data for the SASSy JCMT legacy survey in the QL pipeline

REDUCE_SCAN_EXTENDED_SOURCES_QL Process SCAN data from extended sources in the QL pipeline

REDUCE_SCAN_FAINT_POINT_SOURCES_QL Process SCAN data from faint compact sources in the QL pipeline

REDUCE_SCAN_QL QL process SCAN data

REDUCE_SETUP_QL Process data from a setup observation in the QL pipeline

REDUCE_SKYDIP_QL Process SKYDIP observations in the QL pipeline

Summit-pipeline recipes

REDUCE_FOCUS_SUMMIT Derive focus information in the summit pipeline

REDUCE_FTS_SCAN_SUMMIT Recipe for processing FTS-2 SCAN data in the SUMMIT pipeline

REDUCE_FTS_ZPD_SUMMIT Recipe for processing FTS-2 SCAN data in the SUMMIT pipeline

REDUCE_NOISE_SUMMIT SUMMIT processing NOISE observations

REDUCE_POINTING_SUMMIT Summit processing of pointing observations

REDUCE_POL_STARE_SUMMIT Recipe for processing POL-2 stare data in the SUMMIT pipeline

REDUCE_SASSY_SUMMIT Process data for the SASSy JCMT legacy survey in the SUMMIT pipeline

REDUCE_SCAN_EXTENDED_SOURCES_SUMMIT Process SCAN data from extended sources in the SUMMIT pipeline

REDUCE_SCAN_FAINT_POINT_SOURCES_SUMMIT Process SCAN data from faint compact sources in the SUMMIT pipeline

REDUCE_SCAN_SUMMIT Summit recipe for processing SCAN data

REDUCE_SETUP_SUMMIT Process data from a setup observation in the summit pipeline

REDUCE_SKYDIP_SUMMIT Process SKYDIP observations with the summit pipeline

Miscellaneous

ARRAY_TESTS Co-add and display DREAM/STARE images

BENCHMARK Benchmarking recipe

REDUCE_DARK Process dark frames

G Main Science recipes

FAINT_POINT_SOURCES
alias for REDUCE_SCAN_FAINT_POINT_SOURCES

Description:

This recipe is an alias for the recipe REDUCE_SCAN_FAINT_POINT_SOURCES.

FAINT_POINT_SOURCES_JACKKNIFE
alias for REDUCE_SCAN_FAINT_POINT_SOURCES_JACKKNIFE

Description:

This recipe is an alias for the recipe REDUCE_SCAN_FAINT_POINT_SOURCES_JACKKNIFE.

REDUCE_CLS
alias for REDUCE_SCAN_FAINT_POINT_SOURCES_JACKKNIFE

Description:

This recipe is an alias for the recipe REDUCE_SCAN_FAINT_POINT_SOURCES_JACKKNIFE.

REDUCE_DREAMSTARE
Process DREAM/STARE images

Description:

Input data from the individual subarrays are combined to produce a single mosaic for each time step. Sky removal and extinction correction take place on these images before they are combined into a single Frame image.

The Frame image is calibrated and displayed, before being combined with the Group image in a running average. The noise properties of the new Group image are calculated and logged, and the image searched for sources.

Notes:

- This primitive can not handle time series data.
- The current noise level is stored in log.noise, the positions and fluxes of any sources are stored in log.flux.

Display :

The Frame image is displayed in Gaia window 1. The Group image is displayed in Gaia window 2; its variance is displayed in window 3.

REDUCE_JPS
alias for REDUCE_SCAN_EXTENDED_SOURCES

Description:

This recipe is an alias for the recipe REDUCE_SCAN_EXTENDED_SOURCES.

REDUCE_POINTING alias for REDUCE_SCAN

Description:

This recipe is an alias for the recipe REDUCE_SCAN.

REDUCE_SASSY
Process data for the SASSy project

Description:

This is the recipe for processing data taken for the SCUBA-2 All-Sky Survey (SASSy). The aim of SASSy is to map as much of the sky at 850 um as possible, focussing on the Outer Galaxy covering the Galactic longitude from 120 to 240 degrees. Each input map covers a 2x2 sq degrees

Raw data are passed to the map maker which are processed to produce an image, which is calibrated in mJy/beam. Once all the data for a given target have been processed, the individual images are coadded using inverse-variance weighting. The noise properties of the coadd are calculated and written to a log file, log.noise.

The coadd has a matched-filter applied (to highlight compact sources) and is then passed to the Fellwalker source-finding algorithm to pick out sources at or above the 5-sigma level. A catalogue is written to disk if sources were found.

Notes:

- The image noise and and NEFD are stored in log.noise and log.nefd respectively.
- For large amounts of data this recipe will spend a long time not updating the ORAC-DR output. Check to see that makemap is still processing by running top or ps.
- Alternative configuration parameters for the iterative map-maker may be specified using the recipe parameters outlined below.
- The output map is calibrated in units of mJy/beam.

Available Parameters

The following parameters can be set via the -recpars option:

FINDCLUMPS_CFG

Name of a config file for use with the CUPID findclumps task. The file must exist in the current working directory, $FINDCLUMPS_CONFIG_DIR or $ORAC_DATA_OUT.

MAKEMAP_CONFIG

Name of a config file for use with the SMURF makemap task. The file must exist in the current working directory, $MAKEMAP_CONFIG_DIR or $ORAC_DATA_OUT.

Display :

None.

REDUCE_SCAN
Process SCAN-mode data

Description:

This is the default recipe for processing SCAN data.

Raw data are passed to the map maker which are processed using the default config file (unless the source is a calibrator) to produce a an image. FCFs are derived if the source is a calibrator. This image is calibrated in mJy/beam, displayed and tagged as a reduced product. A coadd is created and displayed after all the individual observations have been processed. The noise properties of the coadd are calculated and written to a log file, log.noise. The NEFD properties of each image produced by makemap and the coadd are written to another log file, log.nefd.

Finally, the CUPID task findclumps is run using the fellwalker algorithm to create a source catalogue.

Notes:

- If given data from multiple sources, each source will be processed in full in turn.
- The noise level and NEFD are stored in log.noise and log.nefd respectively. The noise and NEFD are calculated for each image as well as for the final coadd.
- For large amounts of data this recipe will spend a long time not updating the ORAC-DR window. Check to see that makemap is still processing by running top or ps. (Running with -log sf is recommended.)
- Alternative configuration parameters for the iterative map-maker may be specified using the recipe parameters outlined below.
- Flux conversion factors are derived if the source is a calibrator.
- To reduce the chance of spurious source detections, the group coadd is trimmed to the map size specified in the FITS header before running the source finder. This may not work as intended if the coadd contains maps with different centres.
- The output catalogue, if created will have the extension .FIT. A clump file with suffix _clmp will also be created.
- This recipe can handle data from multiple sources and will correctly coadd maps, taking into account the EXP_TIME and WEIGHTS components.

Available Parameters

The following recipe parameters can be set via the -recpars

option:

CALUNITS
Units in which to calibrate the output map. Can be " BEAM" (mJy/beam), " ARCSEC" (mJy/square arcsecond) or " PW" (map left in pW).

FINDCLUMPS_CFG
Name of a config file for use with the CUPID findclumps task. The file must exist in the current working directory, $FINDCLUMPS_CONFIG_DIR or $ORAC_DATA_OUT.

MAKEMAP_CONFIG
Name of a config file for use with the SMURF makemap task. The file must exist in the current working directory, $MAKEMAP_CONFIG_DIR, $ORAC_DATA_OUT, $ORAC_DATA_CAL or $STARLINK_DIR/share/smurf.

MAKEMAP_PIXSIZE
Size of the pixels in the output map. If not specified, the recipe uses the appropriate default value. Note that the timeseries will be downsampled to match this scale during the map-making process.

MAKEMAP_REF
Name of a reference image (NDF format) to use to define the output pixel grid. The NDF can be either 2D or 3D and the spatial WCS frame will be extracted.

Display :

The Frame image is displayed in Gaia window 1. The Group image is displayed in Gaia window 2; its variance is displayed in window 3.

REDUCE_SCAN_CHECKRMS
Process SCAN-mode data and collect noise/NEFD statistics

Description:

This recipe performs noise and map-making processing on raw time series data in order to derive a number of performance parameters for comparison with predictions from the SCUBA-2 integration time calculator (ITC).

The mean NEP for each subarray is derived from the full timeseries. The average of these values is quoted as the value NEP_AV.

Raw data are passed to the map maker which are processed using the default config file (unless the source is a calibrator) to produce a an image. FCFs are derived if the source is a calibrator. The image is calibrated in mJy/beam using the default FCF.

The recipe calculates the elapsed time of the observation and stores the mean zenith optical depth (at 225 GHz and the current wavelength), line-of-sight transmission and elevation. The NEP data are converted to an NEFD (using the transmission and default FCF) and, using an effective exposure time derived from the ITC, an equivalent image noise.

The noise, NEFD and mean exposure time (per pixel) are derived from the calibrated image. RMS noise and NEFD values are predicted using the ITC and a warning issued if the measured noise exceeds the expected value by more than 20 %.

All values are written to a log file called log.checkrms.

Notes:

- Only default noise processing is supported.
- The user may specify alternate makemap config files, but be aware that the same config will be used for every observation.
- Non-standard PONG map sizes will yield no results from the ITC.
- Undefined values appear as NaN in the log file.
- The recipe does not coadd the data; results are written to the log file for each separate observation.
- Only the reduced map files (one for each observation) are kept at the end of processing.

Available Parameters

All parameters accepted by REDUCE_SCAN may be used in this recipe.

Display :

None.

REDUCE_SCAN_EXTENDED_SOURCES
Process SCAN data from extended sources

Description:

This is the recipe for processing SCAN data for extended (e.g. Galactic) sources. The makemap configuration file is tuned to best deal with such data.

Raw data are passed to the map maker which are processed to produce a Frame image, which is then calibrated and displayed. A new Group image is created and displayed. The noise properties of the new Group image are calculated and written to a log file, log.noise.

Notes:

- The current noise level is stored in log.noise.
- For large amounts of data this recipe will spend a long time not updating the ORAC-DR output. Check to see that makemap is still processing by running top or ps.
- Alternative configuration parameters for the iterative map-maker may be specified using the recipe parameters outlined below.
- The output map is calibrated in units of mJy/arcsec∗∗2.

Available Parameters

The following parameters can be set via the -recpars option:

MAKEMAP_CONFIG

Name of a config file for use with the SMURF makemap task. The file must exist in the current working directory, $MAKEMAP_CONFIG_DIR or $ORAC_DATA_OUT.

Display :

The Frame image is displayed in Gaia window 1. The Group image is displayed in Gaia window 2; its variance is displayed in window 3.

REDUCE_SCAN_FAINT_POINT_SOURCES
Process SCAN data from faint compact sources

Description:

This is the recipe for processing SCAN data for faint compact sources. The makemap configuration file is tuned to best deal with such data, though the user may specify their own.

Raw data are passed to the map maker which are processed using the blank_field config file to produce an image, which is calibrated in mJy/beam, displayed and tagged as a reduced product. The noise is estimated and the NEFD is calculated. Once all the individual observations have been processed, a coadd is created and displayed. This image is processed with a matched filter to enhance the signal-to-noise ratio of point sources. This image is tagged as a reduced product. The noise properties and NEFD for the new Group image are calculated and written to log files. A signal-to-noise ratio image is created from the matched-filtered image.

Finally, the CUPID task findclumps is run using the fellwalker algorithm to create a source catalogue.

Notes:

- If given data from multiple sources, each source will be processed in full in turn.
- The noise level and NEFD are stored in log.noise and log.nefd respectively. The noise and NEFD are calculated for each image as well as for the final coadd (after processing with the matched filter).
- For large amounts of data this recipe will spend a long time not updating the ORAC-DR window. Check to see that makemap is still processing by running top or ps. (Running with -log sf is recommended.)
- Alternative configuration parameters for the iterative map-maker may be specified using the recipe parameters outlined below.
- To reduce the chance of spurious source detections, the group coadd is trimmed to the map size specified in the FITS header before running the source finder. This may not work as intended if the coadd contains maps with different centres.
- The output catalogue, if created will have the extension .FIT. A clump file with suffix _clmp will also be created.
- This recipe can handle data from multiple sources and will correctly coadd maps, taking into account the EXP_TIME and WEIGHTS components.
- This recipe may be called via the shorter name FAINT_POINT_SOURCES.

Available Parameters

The following parameters can be set via the -recpars option:

FINDCLUMPS_CFG
Name of a config file for use with the CUPID findclumps task. The file must exist in the current working directory, $FINDCLUMPS_CONFIG_DIR or $ORAC_DATA_OUT.

MAKEMAP_CONFIG
Name of a config file for use with the SMURF makemap task. The file must exist in the current working directory, $MAKEMAP_CONFIG_DIR, $ORAC_DATA_OUT, $ORAC_DATA_CAL or $STARLINK_DIR/share/smurf.

MAKEMAP_PIXSIZE
Size of the pixels in the output map. If not specified, the recipe uses the appropriate default value. Note that the timeseries will be downsampled to match this scale during the map-making process.

MAKEMAP_REF
Name of a reference image (NDF format) to use to define the output pixel grid. The NDF can be either 2D or 3D and the spatial WCS frame will be extracted.

Display :

The Frame image is displayed in Gaia window 1. The Group image is displayed in Gaia window 2; its variance is displayed in window 3.

REDUCE_SCAN_FAINT_POINT_SOURCES_JACKKNIFE
Process blank field data with a jack-knife-based method

Description:

Process SCAN data and use a jack-knife method to remove residual low-spatial frequency noise plus create an optimal matched-filtered output map. The recipe proceeds as follows:

- Each observation is processed twice using the specified parameters, the second time with an artificial point source added to the timeseries, to create a signal map and an effective PSF image.
- These images are coadded (like with like) to produce a total signal map and a total effective PSF image. The amplitude of the source in the effective PSF image is compared with the input value to assess the effect the map-making process has on a point source. This ratio is used to scale the FCF later when calibrating the data.
- The observations are divided into two groups which are coadded separately. These coadds are subtracted from one another to create the jack-knife map.
- The angular power spectrum of the jack-knife map (which should consist purely of noise) is calculated and used to remove residual low-spatial frequency noise from the signal map and the effective PSF. This is the so-called whitening step (because it produces a map which has a noise power spectrum that is white).
- The data are calibrated in mJy/beam using a corrected FCF.
- The whitened signal map is processed with a matched filter using the whitened PSF image as the PSF.
- The jack-knife map is also whitened and processed with the matched filter. This map should consist purely of noise.
- Signal-to-noise ratio maps are created for the filtered versions of the signal map and the jack-knife map.

The outcome (the match-filtered whitened signal map) should be the optimal map with white noise properties. This is the map to be used for science goals.

This recipe generates a large number of output files. There will be two for each observation (the signal map and the effective PSF map), a total signal coadd, an effective PSF coadded from all the individual effective PSF maps, a jackknife map, a whitened signal map, a calibrated whitened signal map, a matched-filtered calibrated whitened signal map and a signal-to-noise ratio map created from it.

Notes:

- This recipe should only be given data for a single source, and a single field.
- An even number of observations will be used to create the jack-knife map.

- Alternative configuration parameters for the iterative map-maker may be specified using the recipe parameters outlined below.
- This recipe may be called via the shorter name FAINT_POINT_SOURCES_JACKKNIFE.
- For large amounts of data this recipe will spend a long time not updating the ORAC-DR window. Check to see that makemap is still processing by running top or ps. (Running with -log sf is recommended.)

Available Parameters

The following parameters can be set via the -recpars option:

FAKEMAP_CONSTSNR
A flag to scale the gaussian in amplitude by the square-root of the number of maps to be created by the recipe so that the signal-to-noise ratio of the final map-filtered PSF is independent of the number of observations. May be useful for comparing results with different numbers of files. Default is 0 (use the given amplitude).

FAKEMAP_FWHM
FWHM (in arcsec) of a gaussian to add to the timeseries. Default is to use the appropriate telescope main beam FWHM.

FAKEMAP_OFFSET
The offsets (in arcsec) to apply to the fake source added to the timeseries. If one value is given, it will be used for both axes. Default is to apply no shift.

FAKEMAP_SCALE
Amplitude of the fake source (in Jy/beam) added to the timeseries to assess the map-making response to a point source. Default is 10/50 Jy/beam at 850/450 um respectively.

JACKKNIFE_METHOD
Method for creating jack-knife map. May be alternate to use every other file to create the two halves, or half to use the first N/2 files (by date) for one half of the jack-knife and the remainder for the other. Default is alternate.

MAKEMAP_CONFIG
Name of a config file for use with the SMURF makemap task. The file must exist in the current working directory, $MAKEMAP_CONFIG_DIR, $ORAC_DATA_OUT, $ORAC_DATA_CAL or $STARLINK_DIR/share/smurf.

MAKEMAP_PIXSIZE
Pixel size in arcsec for the output map. Default is wavelength dependent (4 arcsec at 850 um, 2 arcsec at 450 um). Note that the timeseries will be downsampled to match this scale during the map-making process.

PSF_BOX
Size of square region (in pixels) use to define effective PSF.

STATS_COMP
Name of component to use when determining the threshold level. Default is texp (the EXP_TIME component).

STATS_ESTIMATOR
Statistical estimator to use to determine threshold level. May be max, mean, median, or min. Default is median.

STATS_THRESH
Threshold multiplier - the threshold will be this value multiplied by the estimator. Default is 0.5 if using the exposure time, 1 otherwise.

WHITEN_BOX
Size of the region used to calculate the angular power spectrum for removing residual low-frequency noise in the data. Default is a square region bounded by the noise being less than twice the minimum value.

WHITEN_ESTIMATOR
Statistical estimator to determine the threshold level to define the size of the whitening region. May be MIN, MEAN or MEDIAN. Default is MIN (see WHITEN_BOX).

WHITEN_THRESH
The threshold multiplier at which to define the size of the whitening region. Default is 2 (see WHITEN_BOX).

Display :

No results are displayed.

REDUCE_SCAN_FAKEMAP
Process SCAN data with existing map data added

Description:

This recipe processes SCAN data with data from an existing map added to the input time-series. Raw data for each observation are passed to the map maker which are processed to produce an image, which is then calibrated. The individual images are coadded using inverse-variance weighting.

The user must provide the name of a template input map from which a fake map is created with the pixel bounds which match those of the map created with SCUBA-2 data alone. The input map may be shifted on the sky relative to its nominal centre position to avoid overlap with sources in the SCUBA-2 data, and may be regridded to match the pixel size of the output map.

Alternatively, the recipe can optionally create a Gaussian of a given FWHM and unit amplitude, though only at the map centre (in pixel coordinates).

Notes:

- For large amounts of data this recipe will spend a long time not updating the ORAC-DR output. Check to see that makemap is still processing by running top or ps.
- Alternative configuration parameters for the iterative map-maker may be specified using the recipe parameters outlined below.

Available Parameters

The following parameters can be set via the -recpars option:

FAKEMAP_FWHM

FWHM in arcsec of Gaussian to use as input map. Only used if FAKEMAP_MAP is unspecified.

FAKEMAP_MAP

Name of the map to add to the raw timeseries. The file must be an NDF and exist in the current working directory or $ORAC_DATA_OUT.

FAKEMAP_SCALE

The value by which to scale the input map before adding to the timeseries. Default is 1.0 (no scaling). Equivalent of the makemap parameter fakescale.

FAKEMAP_REGRID

A flag to denote whether the fake map should be regridded to the same pixel scale as the output map. Default is 0 (false).

FAKEMAP_OFFSET
RA, Dec offsets in arcsec which specify how much the map coordinates should be adjusted before adding to the map. If only one value is given, the same will be used for both axes. Default is 0,0 (no shift).

MAKEMAP_CONFIG
Name of a config file for use with the SMURF makemap task. The file must exist in the current working directory, $MAKEMAP_CONFIG_DIR or $ORAC_DATA_OUT.

Display :

None.

REDUCE_SCAN_ISOLATED_SOURCE
Process SCAN data an isolated source

Description:

This is a recipe for processing SCAN data for a single bright, isolated source at the tracking position.

Raw data are passed to the map maker which are processed to produce a Frame image, which is then calibrated and displayed. A new Group image is created and displayed. The noise properties of the new Group image are calculated and written to a log file, log.noise.

This recipe will attempt to register the individual images (i.e. align the bright central source in all observations) before coadding. If the source isn' t detected well enough to do this it will continue on, with a warning in the log.

Notes:

- The current noise level is stored in log.noise.
- For large amounts of data this recipe will spend a long time not updating the ORAC-DR output. Check to see that makemap is still processing by running top or ps.
- Alternative configuration parameters for the iterative map-maker may be specified using the recipe parameters outlined below.

Available Parameters

The following parameters can be set via the -recpars option:

MAKEMAP_CONFIG

Name of a config file for use with the SMURF makemap task. The file must exist in the current working directory, $MAKEMAP_CONFIG_DIR or $ORAC_DATA_OUT.

Display :

The Frame image is displayed in Gaia window 1. The Group image is displayed in Gaia window 2; its variance is displayed in window 3.

REDUCE_SCAN_JSA_PUBLIC
Form observation tiles for the public co-add

Description:

This is a recipe to process SCUBA-2 SCAN data to create per-observation JSA (HEALPix) tiles. The map maker is called with the JSATILES parameter enabled and with the generic CONFIG file suitable for the JSA public co-added products.

It is based on, and performs the first few steps of, the REDUCE_SCAN recipe but with the adjustments mentioned above. Please see the documentation for that recipe for further information.

REDUCE_SONS
Process data for the SONS project

Description:

This is the recipe for processing data taken for the SCUBA-2 Survey of Nearby Stars (SONS).

This recipe processes SCAN data, making a map from files that meet a given noise criterion, and uses a jack-knife method to remove residual low-spatial frequency noise and create an optimal matched-filtered output map. The recipe proceeds as follows:

- The noise properties for each 30-second subscan are calculated, and those files which exceed 1.5 times the mean are excluded from the map-making stage.
- Each observation is processed twice using the specified parameters, the second time with an artificial point source added to the timeseries, to create a signal map and an effective PSF image.
- These images are coadded (like with like) to produce a total signal map and a total effective PSF image. The amplitude of the source in the effective PSF image is compared with the input value to assess the effect the map-making process has on a point source. This ratio is used to scale the FCF later when calibrating the data.
- The observations are divided into two groups which are coadded separately. These coadds are subtracted from one another to create the jack-knife map.
- The angular power spectrum of the jack-knife map (which should consist purely of noise) is calculated and used to remove residual low-spatial frequency noise from the signal map and the effective PSF. This is the so-called whitening step (because it produces a map which has a noise power spectrum that is white).
- The data are calibrated in mJy/beam using a corrected FCF.
- The whitened signal map is processed with a matched filter using the whitened PSF image as the PSF.
- The jack-knife map is also whitened and processed with the matched filter. This map should consist purely of noise.
- Signal-to-noise ratio maps are created for the filtered versions of the signal map and the jack-knife map.

The outcome (the match-filtered whitened signal map) should be the optimal map with white noise properties. This is the map to be used for science goals.

This recipe generates a large number of output files. There will be two for each observation (the signal map and the effective PSF map), a total signal coadd, an effective PSF coadded from all the individual effective PSF maps, a jackknife map, a whitened signal map, a calibrated whitened signal map, a matched-filtered calibrated whitened signal map and a signal-to-noise ratio map created from it.

Notes:

- This recipe should only be given data for a single source, and a single field.
- An even number of observations will be used to create the jack-knife map.
- Alternative configuration parameters for the iterative map-maker may be specified using the recipe parameters outlined below.
- This recipe may be called via the shorter name FAINT_POINT_SOURCES_JACKKNIFE.
- For large amounts of data this recipe will spend a long time not updating the ORAC-DR window. Check to see that makemap is still processing by running top or ps. (Running with -log sf is recommended.)

Available Parameters

The following parameters can be set via the -recpars option:

FAKEMAP_CONSTSNR
A flag to scale the gaussian in amplitude by the square-root of the number of maps to be created by the recipe so that the signal-to-noise ratio of the final map-filtered PSF is independent of the number of observations. Default is 1.

FAKEMAP_FWHM
FWHM (in arcsec) of a gaussian to add to the timeseries. Default is to use the appropriate telescope main beam FWHM.

FAKEMAP_OFFSET
The offsets (in arcsec) to apply to the fake source added to the timeseries. If one value is given, it will be used for both axes. Default is to apply no shift.

FAKEMAP_SCALE
Amplitude of the fake source (in Jy/beam) added to the timeseries to assess the map-making response to a point source. Default is 10/50 Jy/beam at 850/450 um respectively.

MAKEMAP_CONFIG
Name of a config file for use with the SMURF makemap task. The file must exist in the current working directory, $MAKEMAP_CONFIG_DIR, $ORAC_DATA_OUT, $ORAC_DATA_CAL or $STARLINK_DIR/share/smurf.

MAKEMAP_PIXSIZE
Pixel size in arcsec for the output map. Default is wavelength dependent (4 arcsec at 850 um, 2 arcsec at 450 um). Note that the timeseries will be downsampled to match this scale during the map-making process.

PSF_BOX
Size of square region (in pixels) use to define effective PSF.

SUBSCAN_ESTIMATOR
Estimator for calculating the noise properties of each subscan. Supported values are WTNEP (not for METHOD=MAP), MEAN, MEDIAN and SIGMA (METHOD=MAP only). Default is SIGMA (or MEDIAN if METHOD is not MAP).

SUBSCAN_METHOD
Method for determining the noise properties. Supported values are FREQHI, FREQLO and MAP. Default is MAP.

WHITEN_BOX
Size of the region used to calculate the angular power spectrum for removing residual low-frequency noise in the data. Default is a square region bounded by the noise being less than twice the minimum value.

Display :

None.

H FTS-2 recipes

REDUCE_FTS_FOCUS
Recipe for processing FTS-2 focus observations

Description:

This is a recipe for processing focus observations when FTS-2 is in the beam.

Notes:

This recipe is the same as the standard REDUCE_FOCUS recipe except that it only handles timeseries data because it applies masking to it, and it does not try to calibrate the data.

Display :

None.

REDUCE_FTS_IMAGE_FOCUS
Recipe for processing FTS-2 image focus observations

Description:

This is a recipe for processing focus observations when FTS-2 is in the beam using the image port.

Notes:

This recipe is the same as the standard REDUCE_FOCUS recipe except that it only handles timeseries data because it applies masking to it, and it does not try to calibrate the data.

Display :

None.

REDUCE_FTS_IMAGE_POINTING
Recipe for processing FTS-2 image pointing observations

Description:

This is a recipe for processing pointing observations when FTS-2 is in the beam using the image port.

Notes:

This recipe is the same as the standard REDUCE_POINTING recipe except that it only handles timeseries data because it applies masking to it, and it does not try to calculate an FCF.

Display :

None.

REDUCE_FTS_POINTING
Recipe for processing FTS-2 pointing observations

Description:

This is a recipe for processing pointing observations when FTS-2 is in the beam.

Notes:

This recipe is the same as the standard REDUCE_POINTING recipe except that it only handles timeseries data because it applies masking to it, and it does not try to calculate an FCF.

Display :

None.

REDUCE_FTS_SCAN
Recipe for processing FTS-2 SCAN data

Description:

This is a basic recipe for processing FTS-2 SCAN data. All processing is done via the FTS-2 applications in the SMURF package.

The input data are split into separate scans and then processed to produce spectra.

Notes:

None.

Available Recipe Parameters

FTS_STAGE_CENTER
Center position of moving mirror travel, usually read from the FTS_CNTR FITS header.

FTS_SCANDIR_ZPD
Indicates whether to use scan direction-specific ZPD measurements.

FTS_WN_LBOUND
Lower wavenumber bound.

FTS_WN_UBOUND
Upper wavenumber bound.

Display :

kapview window 1 region 2 A representative spectrum. This is currently simply the first spectrum in the frame.

REDUCE_FTS_SCAN_QL alias for REDUCE_SCAN_QL

Description:

This recipe is an alias for the recipe REDUCE_SCAN_QL.

REDUCE_FTS_SCAN_SUMMIT
alias for REDUCE_FTS_SCAN

Description:

This recipe is an alias for the recipe REDUCE_FTS_SCAN.

REDUCE_FTS_ZPD
Recipe for processing FTS-2 ZPD calibration data

Description:

This recipe processes FTS-2 ZPD calibration data to estimate the location of the ZPD (zero path difference) point. The input data are split into separate scans, baseline-subtracted and labeled with an approximate moving mirror position coordinate system frame. The position of the peak fringe, corresponding to the ZPD position, is estimated for each scan and then averaged for each subarray. Finally, to ensure a smooth ZPD map without gaps, a surface is fitted to the average ZPD positions.

Notes:

The FTS shutter status is normally read from the FTS_SH8D and FTS_SH8C FITS headers. If these are not defined (i.e. in older data) then an attempt is made to read this information from the OCS configuration XML (which of course indicates the requested configuration).

Available Recipe Parameters

FTS_STAGE_CENTER
Center position of moving mirror travel, usually read from the FTS_CNTR FITS header.

FTS_SCANDIR_ZPD
Indicates whether to derive scan direction-specific ZPD measurements.

FTS_ZPD_ESTIMATE
Initial estimate of ZPD position.

FTS_ZPD_TOLERANCE
Tolerance in ZPD position.

FTS_ZPD_BASE_OFFSET
Distance to ZPD baselining region.

FTS_ZPD_BASE_WIDTH
Width of ZPD baselining region.

FTS_ZPD_PEAK_HEIGHT
Initial estimate of ZPD peak height.

FTS_ZPD_PEAK_WIDTH
Initial estimate of ZPD peak width.

Display :

kapview window 1 region 1 Initial ZPD position estimate mosaicked image. kapview window 1 region 3 Initial ZPD position estimate histogram. kapview window 1 region 2 Final ZPD position measurement mosaicked image. kapview window 1 region 4 Final ZPD position measurement histogram.

REDUCE_FTS_ZPD_QL
alias for REDUCE_SCAN_QL

Description:

This recipe is an alias for the recipe REDUCE_SCAN_QL.

REDUCE_FTS_ZPD_SUMMIT
alias for REDUCE_FTS_ZPD

Description:

This recipe is an alias for the recipe REDUCE_FTS_ZPD.

I POL-2 recipes

REDUCE_POL_SCAN_SUMMIT
Process POL-2 scan/spin mode data

Description:

This will attempt to just produce a makemap reduction of the data

- - it will not carry out calcqu to get data.

REDUCE_POL_SCAN_QL
alias for REDUCE_SCAN_QL

Description:

This recipe is an alias for the recipe REDUCE_SCAN_QL.

REDUCE_POL_SCAN_SUMMIT
alias for REDUCE_POL_SCAN

Description:

This recipe is an alias for the recipe REDUCE_POL_SCAN.

REDUCE_POL_STARE
Recipe for processing POL-2 stare data

Description:

This recipe reduces POL-2 stare data and generates Q and U images and vectors.

Notes:

There needs to be a way for an I image to be obtained to allow this recipe to properly generate an IQU cube.

REDUCE_POL_STARE_QL alias for REDUCE_SCAN_QL

Description:

This recipe is an alias for the recipe REDUCE_SCAN_QL.

REDUCE_POL_STARE_SUMMIT
alias for REDUCE_POL_STARE

Description:

This recipe is an alias for the recipe REDUCE_POL_STARE.

J Summit Recipes

REDUCE_POINTING_SUMMIT
Process POINTING observations in the summit pipeline

Description:

This recipe processes data from a POINTING observation in the SUMMIT pipeline.

For DREAM/STARE, the images from the individual subarrays are combined, sky emission removed (assuming a simple DC offset) and corrected for extinction.

SCAN-mode data are passed to the iterative map-maker. Fast-ramp flatfield files are processed and stored so the iterative map-maker can use them.

The image is cropped to 150 arcsec on a side, and if the source is a known calibrator, the pipeline calculates FCFs. The map is then calibrated and tagged as a reduced product.

For additional record keeping, the source position is fitted and offsets in Azimuth and Elevation from the nominal (0,0) position are derived and logged. Note, however, that these may not be identical to those derived by the telescope POINTING_FOCUS task.

This recipe is largely the same as REDUCE_POINTING with minor modifications specific to handle data as they are taken.

Notes:

- The pointing offsets, beam size and FCF are written to the log files log.pointing, log.beam and log.fcf.
- Fast-ramp flatfield data are processed and results written to the log file log.flatfield.
- This recipe deals with both the 2-D DA-processed images and the time series data. Primitives which are meant for 2-D images are no-ops for time-series data.
- The pointing offsets are calculated from the centroid position and a fit to the source. They are written into the output file as a JCMT::Pointing extension.

Display :

The coadd is displayed in Gaia window 2; its variance is displayed in window 3.

REDUCE_SASSY_SUMMIT alias for REDUCE_SCAN_SUMMIT

Description:

This recipe is an alias for the recipe REDUCE_SCAN_SUMMIT.

REDUCE_SCAN_EXTENDED_SOURCES_SUMMIT
alias for REDUCE_SCAN_SUMMIT

Description:

This recipe is an alias for the recipe REDUCE_SCAN_SUMMIT.

REDUCE_SCAN_FAINT_POINT_SOURCES_SUMMIT alias for REDUCE_SCAN_SUMMIT

Description:

This recipe is an alias for the recipe REDUCE_SCAN_SUMMIT.

REDUCE_SCAN_ISOLATED_SOURCE_SUMMIT alias for REDUCE_SCAN_SUMMIT

Description:

This recipe is an alias for the recipe REDUCE_SCAN_SUMMIT.

REDUCE_SCAN_SUMMIT
Summit recipe for processing SCAN data

Description:

This recipe processes SCAN data with the iterative map-maker in the summit pipeline. This recipe makes use of a percentage completion parameter which delays the map-making step until a certain proportion of the data exist on disk.

Flatfielded data are passed to the iterative map maker to produce a Frame image, which is then calibrated and displayed. A new Group image is created and displayed. The noise properties of the new Group image are calculated and written to a log file, log.noise.

Control in this recipe is handled by a series of Frame and Group uhdr flags. These are NOCALIB, OBSEND, PERCENT_CMP and TCS_INDEX (Frame) and LAST_INDEX and MAPMADE (Group). The TCS_INDEX and LAST_INDEX values are used to determine if the scan pattern has been completed since the last map was made. OBSEND is used to decide whether or not to create a new Group coadd. The NOCALIB flag is used to bypass the calibration step as raw timeseries data should not be calibrated.

The PERCENT_CMP entry is specified in this recipe as an argument to _PROCESS_SCAN_DATA_. However, it is only used for observations which consist of a single pass of the scan pattern.

Notes:

- The noise level estimated from the current Group file is stored in log.noise.

Display :

The Frame image is displayed in Gaia window 1 (though no image will be displayed until the percentage completion or change in TCS index criteria are satisfied). The Group image is displayed in Gaia window 2; its variance is displayed in window 3.

K Quick Look Recipes

REDUCE_DREAMSTARE_QL
QL processing DREAM/STARE images

Description:

The data are processed completely as they arrive (at 1 second intervals): images from different subarrays are combined, a mean sky level subtracted, then corrected for extinction and calibrated.

The calibrated Frame image is displayed. Creation of a new Group image is deferred until a minimum number of Frame images have been created to deal with any spikes in the Frame images. This number is set to 10 and may be changed: see the NMOS parameter for _MAKE_MOSAIC_GROUP_DESPIKE_ below. The new Group image is displayed and its noise properties are calculated and logged.

Notes:

- This primitive can not handle time series data.
- The number of images to combine into a new Group image is set by the NMOS parameter for _MAKE_MOSAIC_GROUP_DESPIKE_. Smaller numbers will result in more frequent updates of the Group image at the expense of de-spiking and good variance estimation.
- The current noise level is stored in log.noise, the positions and fluxes of any sources are stored in log.flux.

Display :

The Frame image is displayed in Gaia window 1. The Group image is displayed in Gaia window 2; its variance is displayed in window 3.

REDUCE_POINTING_QL
QL processing of POINTING observations

Description:

This recipe processes data from a POINTING observation with the data obtained in Quick-Look mode.

For DREAM/STARE, the images from the individual subarrays are combined, sky emission removed (assuming a simple DC offset) and corrected for extinction.

SCAN-mode data are passed to the iterative map-maker. Fast-ramp flatfield files are processed and stored so the iterative map-maker can use them.

The image is cropped to 150 arcsec on a side and any residual background is removed (if necessary). If the source is a known calibrator, the pipeline calculates FCFs. The image is then calibrated and a flag flag file written to allow the POINTING_FOCUS task at the summit to derive pointing offsets.

For additional record keeping the source position is fitted to derive offsets in Azimuth and Elevation from the nominal (0,0) position. These offsets are written as a JCMT::Pointing extension in the coadd, and to a log file. Note, however, that these may not be identical to those derived by the telescope POINTING_FOCUS task. The recipe also determines the beam size and the flux conversion factor (FCF), which are also written to their respective log files.

Notes:

- The pointing offsets, beam size and FCF are written to the log files log.pointing, log.beam and log.fcf.
- Fast-ramp flatfield data are processed and results written to the log file log.flatfield.
- The pointing offsets are calculated from the centroid position and a fit to the source. They are written into the output file as a JCMT::Pointing extension as an initial guess for the POINTING_FOCUS task.

Display :

The coadd is displayed in Gaia window 2; its variance is displayed in window 3.

REDUCE_SASSY_QL
alias for REDUCE_SCAN_QL

Description:

This recipe is an alias for the recipe REDUCE_SCAN_QL.

REDUCE_SCAN_EXTENDED_SOURCES_QL alias for REDUCE_SCAN_QL

Description:

This recipe is an alias for the recipe REDUCE_SCAN_QL.

REDUCE_SCAN_FAINT_POINT_SOURCES_QL
alias for REDUCE_SCAN_QL

Description:

This recipe is an alias for the recipe REDUCE_SCAN_QL.

REDUCE_SCAN_ISOLATED_SOURCE_QL
alias for REDUCE_SCAN_QL

Description:

This recipe is an alias for the recipe REDUCE_SCAN_QL.

REDUCE_SCAN_QL
QL process SCAN data

Description:

This recipe processes SCAN data from the quick-look pipeline. The emphasis is on monitoring the noise performance of the instrument and each file is treated as a noise observation.

Notes:

The bolometer noise properties are stored in log.bolonoise and index.noise.

Display :

The noise images for each subarray are displayed as a focal-plane mosaic along with a histogram of the noise values.

L Recipes for Non Science Observations

ARRAY_TESTS
Co-add and display DREAM/STARE images

Description:

Basic processing recipe suitable for DREAM/STARE images from a single subarray. This recipe is designed to be used for lab-testing only.

No alignment is done on the images so that they are mosaicked in the pixel frame (this means that if data exists from multiple subarrays, they will be averaged together!). A method to determine the frame-to-frame variance is provided which writes out to a log file called log.noise which can be monitored by the StripChart. The relevant column to monitor is " noise" .

Notes:

- This primitive is not designed to handle time series observing modes.
- If the frames are identical, then the Group mosaicked image with contain " NaN" . In that case, add the argument GENVAR=0 to _MAKE_MOSAIC_FRAME_.
- If there is no variance, " NaN" will be written to log.noise. The stripchart may or may not do something sensible with that.
- The Group image variance is written to log.noise for monitoring with the StripChart.

Display :

The Frame image is displayed in Gaia window 1. The Group image is displayed in Gaia window 2; its variance is displayed in window 3.

ASSESS_DATA_NOISE
Calculate noise properties of SCUBA-2 data

Description:

This recipe is designed to provide observers with a simple assessment of the noise properties of their data. It does this by calculating the noise for either the first on-sky data file (default) or for the entire time stream. The results are passed through the same quality assurance (QA) checks as performed by the quick-look (QL) pipeline running at the telescope.

The processing is controlled by the recipe parameters described below.

Notes:

None.

Available Parameters

The following recipe parameters can be set via the -recpars

option:

NOISE_CALC
Type of noise calculation to perform. May be full, which calculates the noise properties of the entire time stream or quick to use only the first on-sky subscan (30 seconds).

NOISE_CFG
The name of a config file to use when calculating the noise properties. The default is to use the standard noise config but the user is advised to supply the same config used in the map-making process.

NOISE_FREQLO
The frequency at which to calculate the low-frequency noise. Default is 0.5 Hz.

NOISE_FREQRANGE
A pair of numbers indicating the frequency range (in Hz) over which the noise is to be calculated. Default is 2,10.

Display :

The focal-plane mosaic for the noise and NEP are displayed, scaled between 0 and the relevant spec defined by the quality assurance parameters. A histogram of noise values is also plotted, with the same upper bound.

BENCHMARK
Benchmarking recipe

Description:

Simple but realistic recipe suitable for benchmarking tests. Processes DREAM/STARE images. The quantity of data should always be the same so as to obtain meaningful comparisons.

No images are displayed. This is purely a timing test for processing a standard quantity of data.

Notes:

- This recipe was originally written to process DREAM/STARE images but has not been updated to handle time series data. Its use is not recommended.

Display :

None.

ENG_NEP
Process NEP measurements made in engineering mode

Description:

This recipe processes NOISE observations taken at various settings for the pixel heater and detector bias. The noise properties are calculated over a given frequency range and the resulting noise-equivalent power (NEP) data for each subarray are combined into a 4-d hypercube of the NEP as a function of bolometer position, heater and detector bias value.

In addition the effective and weighted NEPs are calculated for each subarray and heater/bias setting.

Details are written to log files: the noise properties are written to log.bolonoise as for regular noise observations, while the NEPs are written to log.effnep and log.wtnep.

Notes:

- Not designed to run in real time.
- Bias ramp data are ignored.

Available Parameters

The following parameters can be set via the -recpars option:

NEP_MAX

Maximum NEP to be used in calculating the NEP images (W Hz∗∗-0.5). Default is 2.0e-14 W Hz∗∗-0.5.

NEP_MIN

Minimum NEP to be used in calculating the NEP images (W Hz∗∗-0.5). Default is 2.0e-17 W Hz∗∗-0.5.

NOISE_FREQRANGE

A pair of comma-separated values indicating the lower and upper frequency bounds (in Hz) to be used in the calculation of the noise properties. Default is 2,10 Hz.

Display :

No images are displayed.

ENG_NEP_QL
QL processing NEP measurements made in engineering mode

Description:

This recipe calculates the effective noise-equivalent power (NEP) for each subarray for a range of pixel heater and bias values. The results are written as an image with heater values along the x-axis and bias values along the y-axis.

Notes:

- Bias ramp data are ignored.
- This recipe is not up to date and should not be used at this stage.

Display :

Each image is displayed in a single KAPVIEW window.

REDUCE_DARK
Process dark frames

Description:

This recipes handles dark measurements either processing the dark data to form a dark " frame" (image) for DREAM/STARE observations or making a local copy to pass to the iterative map-maker (SCAN data).

The dark files are stored in the calibration system for later retrieval.

Notes:

- Dark file names are stored in index.dark.

Display :

None.

REDUCE_FASTFLAT
Process fast-ramp flatfield data

Description:

Process fast-ramp flatfield data associated with science observations. The fast-ramp flatfields for a given science observation are reduced in turn and compared with one another to see how much the flatfield solution changes over the duration of an observation.

See REDUCE_FLATFIELD for further details.

Notes:

- This recipe only works with raw time-series data.

Display :

The results for each subarray are displayed in a separate Kapview window. The results displayed are:

- Current responsivity solution (top left panel)
- Previous responsivity solution, using same colour scale as the current solution (top right panel)
- Percentage change in responsivities (bottom right panel)
- Histograms of current and previous responsivities, displayed over the same range (left and right bottom left panels respectively)

REDUCE_FLATFIELD
Process flatfield measurements

Description:

This recipe processes data from a FLATFIELD observation and computes a new flatfield solution for each subarray. A responsivity map is created from this solution, and compared with the responsivity map derived from the existing solution. The results are displayed numerically and graphically (with each subarray in a separate window) for user assessment.

In addition a 3-d cube of the responsivity history is created for each subarray (provided a sufficient number of solutions exist), allowing stable bolometers to be identified for heater tracking.

The flatfield solution is kept on disk and stored in the calibration system; all other files created are deleted.

Notes:

- This recipe only works with raw time-series data.
- Flatfield file names are stored in index.flat.

Display :

The results for each subarray are displayed in a separate Kapview window. The results displayed are:

- Current responsivity solution (top left panel)
- Previous responsivity solution, using same colour scale as the current solution (top right panel)
- Percentage change in responsivities (bottom right panel)
- Histograms of current and previous responsivities, displayed over the same range (left and right bottom left panels respectively)

REDUCE_FOCUS
Recipe for deriving focus information

Description:

This recipe processes data for FOCUS observations, creating images for each SMU position. When enough data exist - defined as the existence of a minimum number of images for the last of the expected focus positions - the images for each focus position are combined and a data cube is created with SMU offset in mm as the third axis.

Once the cube has been created, a flag file is written to indicate to the telescope POINTING_FOCUS task that processing is complete. The recipe also makes its own estimate of the best-fit focus position by fitting a parabola to the fitted peak flux densities at each SMU position. The solution is written to a log file, log.focus.

Notes:

- The best-fit focus position is written to log.focus.
- The minimum number of images referred to above is defined as the mean number of images obtained for each of the preceding focus positions.
- No data are displayed until the cube of focus images has been created.
- The bright_compact config file is used.

Display :

The cube is displayed in a Kapview window with the image at each SMU position displayed in a separate panel.

REDUCE_FOCUS_QL
process focus observations in the quick-look pipeline

Description:

This recipe processes data for FOCUS observations, creating images for each SMU position. When enough data exist - defined as the existence of a minimum number of images for the last of the expected focus positions - the images for each focus position are combined and a data cube is created with SMU offset in mm as the third axis.

Once the cube has been created, a flag file is written to indicate to the telescope POINTING_FOCUS task that processing is complete. The recipe also makes its own estimate of the best-fit focus position by fitting a parabola to the fitted peak flux densities at each SMU position. The solution is written to a log file, log.focus.

This primitive is largely the same as REDUCE_FOCUS with minor modifications to handle data as they are taken.

Notes:

- The best-fit focus position is written to log.focus.
- The minimum number of images referred to above is defined as the mean number of images obtained for each of the preceding focus positions. The recipe may fail to produce a cube if the pipeline does not receive enough data, even once the observation has ended. This is because the OBSEND flag in the FITS header will only be true for the final subscan and thus the QL pipeline may not see it.
- No data are displayed until the cube of focus images has been created.
- The bright_compact config file is used.
- See also REDUCE_FOCUS.

Display :

The cube displayed in a Kapview window with the image at each SMU position displayed in a separate panel.

REDUCE_FOCUS_SUMMIT
derive focus information in the summit pipeline

Description:

This recipe processes data for FOCUS observations, creating images for each SMU position. When enough data exist - defined as the existence of a minimum number of images for the last of the expected focus positions - the images for each focus position are combined and a data cube is created with SMU offset in mm as the third axis.

Once the cube has been created, a flag file is written to indicate to the telescope POINTING_FOCUS task that processing is complete. The recipe also makes its own estimate of the best-fit focus position by fitting a parabola to the fitted peak flux densities at each SMU position. The solution is written to a log file, log.focus.

This primitive is largely the same as REDUCE_FOCUS with minor modifications to handle data as they are taken.

Notes:

- The best-fit focus position is written to log.focus.
- The minimum number of images referred to above is defined as the mean number of images obtained for each of the preceding focus positions. The recipe may fail to produce a cube if the pipeline does not receive enough data, even once the observation has ended. This is because the OBSEND flag in the FITS header will only be true for the final subscan and thus the summit pipeline may not see it.
- No data are displayed until the cube of focus images has been created.
- The bright_compact config file is used.
- See also REDUCE_FOCUS.

Display :

The cube is displayed in a Kapview window with the image at each SMUR position displayed in a separate panel.

REDUCE_NOISE
Process NOISE observations

Description:

The recipe calculates the average power spectrum for each array, displaying each in a separate window, before calculating the white noise properties (and noise equivalent power). The noise properties are written to a log file, log.bolonoise. The array mapping speed is calculated and written to a log file called log.mapspeed_SUBARRAY.

The results are displayed in a Kapview window.

Notes:

- This recipe only works with raw time-series data.
- Noise properties are written to log.bolonoise, and mapping speeds are written to log.mapspeed_SUB where SUB is the subarray.

Available Parameters

The following parameters can be set via the -recpars option:

RESIST_CFG

Name of an alternative list of resistor values used when calculating the flatfield. File must be in ORAC_DATA_OUT.

Display :

One Kapview window is used to display the average power spectra for each subarray. The focal-plane noise map and histogram are shown in a separate Kapview window, along with maps of the NEP and the percentage-change in the noise.

REDUCE_NOISE_PHOTON
determine photon noise contribution to total noise

Description:

This recipe derives the photon noise contribution by subtracting in quadrature noise measurements made with the shutter open and closed. The initial dark in the observation is used to estimate the dark noise, and the same number of samples in the first on-sky data are used to calculate the sky noise. The photon noise-equivalent-power (NEP) is then given by

NEP_ph = sqrt(NEP_sky∗∗2 - NEP_dark∗∗2)The image for each subarray is derived separately.

The results are written to a log file called log.photnoise

Notes:

This recipe only works with raw time-series data and is designed to be used in an offline mode.

Available Parameters

The following parameters can be set via the -recpars option:

FLATSNR
Signal-to-noise ratio to use in the flatfield calculation.

NEP_CLIP
A comma-separated pair of numbers to indicate the low and high clipping levels for the NEP. If only one is given, it is assigned to the nepcliplow parameter.

NOI_CLIP
A comma-separated pair of numbers to indicate the low and high clipping levels for the noise. If only one is given, it is assigned to the noicliplow parameter.

NOISE_CFG
Name of an alternative config file used by calcnoise. File must be in ORAC_DATA_OUT.

RESIST_CFG
Name of an alternative list of resistor values used when calculating the flatfield. File must be in ORAC_DATA_OUT.

NOISE_SAMPLES
Number of samples to use when calculating the noise properties of on-sky data. Default is to use the same number as in the initial DARK.

Display :

None.

REDUCE_NOISE_QL
QL processing NOISE observations

Description:

Basic processing recipe suitable for NOISE observations in the quick-look pipeline. Each subarray is handled separately. The noise and NEP properties are calculated and displayed.

The outcome of this recipe is a noise image and bad-bolometer mask for each subarray which is stored in the calibration system.

Notes:

This recipe only works with raw time-series data.

Display :

The focal-plane noise map and corresponding histogram are displayed in a Kapview window along with the focal-plane NEP map and the percentage-change in the noise.

REDUCE_NOISE_SUMMIT
Summit processing NOISE observations

Description:

Summit-pipeline recipe for processing NOISE observations. The bolometer power spectra are calculated and the relative power at two (user-specifiable) frequencies is calculated. If this ratio exceeds a critical value, defined by the MASKVAL parameter below, the bolometer is marked as bad.

The outcome of this recipe is a noise image and bad-bolometer mask for each subarray which is stored in the calibration system.

Notes:

- This recipe only works with raw time-series data.
- See the documentation for _CREATE_BAD_BOLO_MASK_ for further parameters which may be specified to determine bad bolometers.
- Bad-bolometer mask file names are stored in index.mask.

Available Parameters

The following parameters can be set via the -recpars option:

BESTBOL_PERCENT

The number of bolometers expressed as a percentage which should be used to create a bad-bolometer mask.

Display :

The average bolometer power spectrum is displayed in a Kapview window. The noise map and corresponding histogram are displayed in a different Kapview window (each subarray is displayed in different window).

REDUCE_SETUP
process data from a setup observation

Description:

A setup observation consists of a fastramp flatfield followed by a dark noise. This recipe is designed to process these data and will write a flag file for the TCS to identify when to analyze the flatfield and noise properties.

Notes:

None.

Display :

The flatfield solution and noise results are displayed in separate Kapview windows

REDUCE_SETUP_QL
process data from a setup observation in the QL pipeline

Description:

A setup observation consists of a fastramp flatfield followed by a dark noise. This recipe is designed to process these data and will write a flag file for the TCS to identify when to analyze the flatfield and noise properties.

Notes:

This is the QL-specific version of the REDUCE_SETUP recipe.

Display :

The flatfield solution and noise results are displayed in separate Kapview windows

REDUCE_SETUP_SUMMIT
process data from a setup observation in the summit pipeline

Description:

A setup observation consists of a fastramp flatfield followed by a dark noise. This recipe is designed to process these data and will write a flag file for the TCS to identify when to analyze the flatfield and noise properties.

Notes:

This is the QL-specific version of the REDUCE_SETUP recipe.

Display :

The flatfield solution and noise results are displayed in separate Kapview windows

REDUCE_SKYDIP
Process SKYDIP observations

Description:

Placeholder recipe for processing skydip observations.

A skydip is a series of open-shutter noise measurements at different elevations (airmasses). This recipe processes each noise measurement and writes the results to log files. See REDUCE_NOISE for further details.

Notes:

- Since no skydip primitives exist, this recipe fills the time by calculating the noise.

Display :

Displays the current noise image(s).

REDUCE_SKYDIP_QL
Process SKYDIP observations with the quick-look pipeline

Description:

Quick-look pipeline recipe for processing skydip observations.

A skydip is a series of open-shutter noise measurements at different elevations (airmasses). This recipe processes each noise measurement and writes the results to log files. See REDUCE_NOISE for further details.

Notes:

Since no specific skydip primitives exist, this recipe fills the time by calculating the noise.

Display :

Displays the current noise image(s).

REDUCE_SKYDIP_SUMMIT
Process SKYDIP observations with the summit pipeline

Description:

SUMMIT-pipeline recipe for processing skydip observations.

A skydip is a series of open-shutter noise measurements at different elevations (airmasses). This recipe processes each noise measurement and writes the results to log files. See REDUCE_NOISE for further details.

Notes:

Since no specific skydip primitives exist, this recipe fills the time by calculating the noise.

Display :

Displays the current noise image(s).

M Obsolete Recipes

REDUCE_POINTING
Process POINTING observations

Description:

This recipe combines images from a POINTING observation, removes sky emission and corrects for extinction before calculating the offsets in Azimuth and Elevation from the nominal (0,0) position. The recipe also determines the beam size and the flux conversion factor (FCF) and writes them to log files.

The image is processed with a matched filter before estimating the pointing offsets, but this output is not retained on disk. The reduced product remains the coadd.

For additional record keeping, the source position is fitted and offsets in Azimuth and Elevation from the nominal (0,0) position are derived and logged. Note, however, that these may not be identical to those derived by the telescope POINTING_FOCUS task.

Notes:

- The pointing offsets, beam size and FCF are written to the log files log.pointing, log.beam and log.fcf.
- This recipe deals with both the 2-D DA-processed images and the time series data. Primitives which are meant for 2-D images are no-ops for time-series data.
- The pointing offsets are calculated from the centroid position and a fit to the source.

Display :

The coadd is displayed in Gaia window 2; its variance is displayed in window 3.