SPHERE Spectro-Polarimetric High-contrast Exoplanet REsearch	REF: **VLT-TRE-SPH-14690-660/1/0** Issue: 1 Version 14 Date: 28 November 2014 Pages: 271

SPHERE

Data Reduction Pipeline Manual

Contr	Man-PA	Sci	Syst	INS	DRH	CPI	IRD	IFS	ZIM
					X				

Prepared by: **Name**: Ole Möller-Nilsson **Institute**: MPIA **Date**: 28-11-2014	*Signature:*
Approved by: **Name**: Alexey Pavlov **Institute**:MPIA **Date**: 28-11-2014	*Signature:*
Released by: **Name**: Markus Feldt **Institute**: MPIA **Date**: 28-11-2014	*Signature:*

Contributors: Alexey Pavlov, Christian Thalmann, Joe Carson and Markus Feldt, MPIA

Contents

Chapter 1

Introduction

1.1 Purpose

The SPHERE pipeline is a subsystem of the VLT Data Flow System (DFS). It is used in two operational environments, for the ESO Data Flow Operations (DFO), and for the Paranal Science Operations (PSO), in the quick-look assessment of data, in the generation of master calibration data, in the reduction of scientific exposures, and in the data quality control. Additionally, the SPHERE pipeline recipes are made public to the user community, to allow a more personalised processing of the data from the instrument.

The purpose of this document is to describe a typical SPHERE data reduction sequence with the SPHERE pipeline. This manual is a complete description of the data reduction recipes offered by the SPHERE pipeline, reflecting the status of the SPHERE pipeline as of 16th Oct 2013 (version 0.13.0).

1.2 Scope of this document

This document describes the data reduction library for the Sphere instrument on VLT. It is part of the deliverables.

The main purpose of this document is to present and explain the data reduction software (in form of a library) for SPHERE in general and IFS, IRDIS and ZIMPOL in particular. The structure and content of this document follows the guidelines set out in the ESO document "Data Flow for VLT/VLTI Instruments Deliverables Specifications" (VLT-SPE-ESO-19000-1618).

The document presented here follows the layout presented in section 4.5.1 of the "Data Flow for VLT/VLTI Instruments Deliverables Specifications" closely with the exception of an added introduction (chapter 1 in this document) and an overview (chapter 2).

The present document describes the design of the data reduction software, including detailed descriptions of algorithms and functions and explains how to reduce SPHERE data with it. Since this is part of an ongoing development process in close contact with manufacture, testing and verification of the SPHERE hardware and instrument design this document describes only a current status and it is unavoidable that there are several details regarding the software design and implementation that can only be considered preliminary.

The instrument and detector calibrations as discussed here assume that the hardware requirements for the various sub-systems and specified in the corresponding documents are met and that there are no unforeseen instrument signatures.

1.3 Acknowledgments

1.4 Change Record

Issue	Rev.	C.S	change	Date	Comment
0	1				Initial Draft
0	2			5-8-2010	IFS AIT Release I
0	3			1-9-2010	IRDIS AIT Release I
0	4			26-11-2010	November (“Virgo”) Release
0	5		Updated/changed recipe descriptions for detector related recipes (dark, ron, gain, flat, ifs_persistence). Added output keywords	9-3-2011	Libra release.
0	6		Added new ZIMPOL recipes. Updated description of wavelength related IFS recipes (spectra_positions, instrument_flat,....)	15-4-2011	Scorpio release.
0	7			5-10-2011	Release for DRH Science meeting (version 0.11.1)
0	8			27-7-2012	PAE internal release
1	0	6 3.x 9.2 9.14 A.1 3.3 3.3 8,9,10	Data description updated Chapter Removed, now sec. 2.1 Updated descritpion of flatfield Deleted IFS RON recipe QC keywords moved to appendix Adapted example description to match PAE data set updated recipe manual sections	16-10-2013	Ready PAE Release
2					

1.5 Applicable Documents

No.	Document name	Document number, Iss./Rev.
AD1	VLT instrumentation software specifications	VLT-SPE-ESO-17212-0001
AD2	Field and Pupil Rotation for the VLT Units	VLT Report No. 63, ESO 1990
AD3	Data Flow for VLT/VLTI Instruments: Deliverables Specification	VLT-SPE-ESO-19000-1618/2.0
AD4	Common Pipeline Library Technical Developers Manual	VLT-MAN-ESO-19500-3349
AD5	Common Pipeline Library User Manual	VLT-MAN-ESO-19500-2720
AD6	IFS Calibration Plan	VLT-PLA-SPH-14690-0200
AD7	Data Flow for VLT/VLTI Instruments: Deliverables Specification	VLT-SPE-ESO-19000-1618/2.0
AD8	IRDIS Data Reduction Library Design	VLT-TRE-SPH-14690-351
AD9	SPHERE Science Analysis Report	VLT-TRE-SPH-14609-235

1.6 Reference Documents

No.	Document name	Document number, Iss./Rev.
RD1	Efficient algorithms for robust feature matching.	D. M. Mount, N.S. Netanyahu, J. Le Moigne, Pattern Recognition vol. 32 (1999) pp. 17-38.
RD2	Frame combination techniques for ultra-high-contrast imaging	Carson et al. 2008, SPIE, 7014E, 115C
RD3	IRACproc: a software suite for processing and analyzing Spitzer/IRAC data	Schuster et al. 2006, SPIE, 6270E, 65S
RD4	HST Dither Handbook	Koekemoer et al. 2000, [Baltimore: STScI]
RD5	Frame combination using Drizzle	Fruchter, A.S and Hook, R.N 1997 in Proc. SPIE, Vol. 3164
RD6	Euro3D Format	M. Kissler-Pattig et al., Issue 1.2, May 2003
RD7	IFS Simulation report	VLT-TRE-SPH-14690-0195
RD8	IFS Calibration Plan	VLT-PLA-SPH-14690-0200
RD9	Gasgano User's Manual.	http://www.eso.org/gasgano/ VLT-PRO-ESO-19000-1932. 13, 15, 19, 28
RD10	SPHERE User Manual	VLT-MAN-SPH-14690-0430
RD11	SPHERE DRH Test Plan and Report	VLT-PLA-SPH-14690-0659/2/0

1.7 Acronyms

Acronym	Meaning	Mathematical representation
ANSI-C	The standardized programming language C	
API	Advanced Programming Interface	
CCD	Charge Coupled Device	
CFITSIO	A library for accessing FITS files in C	
CPL	Common Pipeline Library	
CVS	Concurrent Version System	
DBI	Double Imaging Mode	
DC	Dark current	$DC(x, y)$
DF	Detector flat field – the pixel response to an input signal.	$DF(x, y)$
DIT	Detector Integration Time	
DRH	Data Reduction Handling	
DPI	Double Polarization Imaging	
DPR	Data Product	
ESO	European Southern Observatory	
FDR	Final Design Review	
FoV	Field of View	
FPN	Fixed Pattern Noise	$FPN(x, y)$
GSL	Gnu Scientific Library	
HDU	Hierarchical Detector Unit	
HST	Hubble Space Telescope	
HWP	Half-wave plate	
IF	Instrument flat field – the lenslet response to an input signal	$IF(x, y; \Delta x, \Delta y, \lambda)$
IFS	The SPHERE integral field spectrograph instrument	
IFU	Integral Field Unit.	
IRDIS	Sphere imaging instrument	
LRS	Low Resolution Spectroscopy	
LDT	Lenslet Description Table	
MRS	Medium Resolution Spectroscopy	
PAE	Preliminary Acceptance Europe	
PDR	Preliminary Design Review	
PDT	Pixel Description Table	
PRO	Product (FITS keywords)	
PSF	Point Spread Function	
QC	Quality Control	
RON	Read out noise	RON
SVN	“Subversion”– a revision control management system	
TBC	To be confirmed	
TBD	To be decided	
TF	Telescope flat field – the flat field response of the telescope	$TF(x, y; \Delta x, \Delta y, \lambda)$
VLT	Very Large Telescope	
ZPL	Zurich Imaging POLarimeter	

Chapter 2

Overview

In collaboration with instrument consortia, the Data Flow Systems Department (DFS) of the Data Management and Operation Division is implementing data reduction pipelines for the most commonly used VLT/VLTI instrument modes. These data reduction pipelines have the following three main purposes:

- Data quality control: pipelines are used to produce the quantitative information necessary to monitor instrument performance.
- Master calibration product creation: pipelines are used to produce master calibration products (e.g. combined dark frames, super-flats, wavelength dispersion solutions).
- Science product creation: using pipeline-generated master calibration products, science products are produced for the supported instrument modes.

The accuracy of the science products is limited by the quality of the available master calibration products and by the algorithmic implementation of the pipelines themselves. In particular, adopted automatic reduction strategies may not be suitable or optimal for all scientific goals.

Instrument pipelines consist of a set of data processing modules that can be called from the command line, from the automatic data management tools available on Paranal or from Gasgano. ESO offers two front-end applications for launching pipeline recipes, Gasgano [14] and EsoRex, both included in the pipeline distribution. These applications can also be downloaded separately from http://www.eso.org/gasgano and http://www.eso.org/cpl/esorex.html. An illustrated introduction to Gasgano is provided in the "Quick Start" Section of this manual.

The SPHERE instrument and the different types of SPHERE raw frames and auxiliary data are described in Sections 3, 6.1, and 7. A brief introduction to the usage of the available reduction recipes using Gasgano or EsoRex is presented in Section 4. In section 5 we advice the user about known data reduction problems. An overview of the data reduction, what are the input data, and the recipes involved in the calibration cascade is provided in Section 8. More details on what are inputs, products, quality control measured quantities, and controlling parameters of each recipe is given in Section 9 (IRDIS), 10 (IFS) and 11 (ZIMPOL). In Section 12 the installation of the SPHERE pipeline recipes is described together with a simple example data reduction of SPHERE data using EsoRex.

2.1 The SPHERE Instrument: IFS, IRDIS and ZIMPOL

In addition to this document, there is the SPHERE user manual [RD10] which gives a brief indtroduction to the instrument, as well as a general description of the available observing instruments and modes, and details about setting up actual observations.

Chapter 3

Quick Start

This section describes the most immediate usage of the SPHERE pipeline recipes.

3.1 SPHERE pipeline recipes

The currently implemented SPHERE recipes are:

IFS:

sph_ifs_master_dark: creation of master dark frame

sph_ifs_master_detector_flat: creation of master detector flat frame from frames taken with the internal flat calibration lamps

sph_ifs_spectra_positions: assignment of spectral regions in standard dithering position and creation of pixel description table

sph_ifs_instrument_flat: creation of total master flat field (including instrument and detector)

sph_ifs_wave_calib: assignment of wavelengths to pixels using wavelength calibration frames taken with calibration lasers

sph_ifs_science_dr: data reduction recipe including full calibration, extraction of wavelength cube

sph_ifs_simple_adi: simple angular differential imaging (ADI) on processed IFS cubes

sph_ifs_spec_deconv: speckle fitting and removal on processed IFS science cubes

sph_ifs_astrometry: perform astrometry calibration of frames, measuring rotation and plate scale

sph_ifs_atmospheric: measure the atmospheric contribution to science data

sph_ifs_std_phot: perform the standard photometry calibration and measure the photometric zero-point

sph_ifs_flux_calib: perform the flux calibration taking the coronagraph into account

sph_ifs_sky_flat: create a sky flat field

sph_ifs_sky_cal: create a sky calibration frame

sph_ifs_gain: measurement of the gain

sph_ifs_detector_persistence: measurement of the detector persistence

sph_ifs_cal_background: measurement of the instrument background

sph_ifs_distortion_map: measurement of the instrument optics induced distortion

sph_ifs_dithering_effects: measure the effect of dithering on the wavelength calibration

IRDIS:

sph_ird_master_dark: creation of master dark frame

sph_ird_instrument_flat: creation of master flat field

sph_ird_science_dbi: data reduction recipe including full calibration and ADI and SDI for DBI mode

sph_ird_andromeda: high level data reduction recipe for differential imaging using the ANDROMEDA algorithm

sph_ird_loci: high level data reduction recipe for differential imaging using the LOCI algorithm

sph_ird_science_imaging: data reduction recipe including full calibration and simple ADI for classical imaging mode

sph_ird_wave_calib: wavelength calibration for IRDIS long slit spectroscopy modes (MRS and LRS)

sph_ird_gain: measurement of the gain

sph_ird_detector_persistence: measurement of the detector persistence

sph_ird_distortion_map: measurement of the overall distortion (instrument and detector)

sph_ird_science_spectroscopy: data reduction of spectroscopic science data

sph_ird_ins_pol: measurement of the instrument induced polarization

sph_ird_ins_pol_eff: measurement of the instrument polarization efficiency

sph_ird_spectra_resolution: measurement of the spectra resolution in wavelength and spatial directions

sph_ird_astrometry: Astrometric determination of plate scale and reference rotation angle

sph_ird_atmospheric: Atmospheric master calibration for spectra modes

sph_ird_flux_calib: Flux calibration

sph_ird_ins_throughput: Measurement of the instrument throughput

sph_ird_pol_zpa_eff: IRDIS polarisation zero position and efficiency

sph_ird_psf_reference: creation of a PSF reference image

sph_ird_sky_background: determination of the sky background

sph_ird_spectra_resolution: resolution measurements for spectra mode

sph_ird_star_center: the field centre determination

sph_ird_tel_pol_offset: telescope polarisation offset

sph_ird_tff: telescope flat field and large scale structure

ZIMPOL:

sph_zpl_preproc: Pre-processing of the zimpol raw data, polarimetric modes (utlity recipe).

sph_zpl_preproc_imaging: Pre-processing of the zimpol raw data, imaging mode (utility recipe).

sph_zpl_master_bias: Create master bias, polarization modes.

sph_zpl_master_bias_imaging: Create master bias, imaging mode.

sph_zpl_master_dark: Create master dark, polarization modes.

sph_zpl_master_dark_imaging: Create master dark, imaging mode.

sph_zpl_intensity_flat: Create intensity flat field, polarimetric modes.

sph_zpl_intensity_flat_imaging: Create intensity flat field, imaging mode.

sph_zpl_polarization_flat: Create polarization flat field, polarimetric modes.

sph_zpl_modem_efficiency: Create modem efficiency, polarimetric modes.

sph_zpl_aoc_efficiency: Measure AOC efficiency, polarimetric modes (not currently applicable).

sph_zpl_aoc_offset: Measure AOC polarization offset, polarimetric modes (not currently applicable).

sph_zpl_aoc_crosstalk: Measure AOC crosstalk, polarimetric modes (not currently applicable).

sph_zpl_zimpol_crosstalk: Measure zimpol crosstalk, polarimetric modes (not currently applicable).

sph_zpl_basic_polarization: Calibrate frames, polarimetric modes (utility recipe).

sph_zpl_basic_collapse_polarization: Calibrate and collapse frames, polarimetric modes (utility recipe).

sph_zpl_basic_imaging: Calibrate frames, imaging mode (utility recipe).

sph_zpl_basic_collapse_imaging: Calibrate and collapse frames, imaging mode (utility recipe).

sph_zpl_science_p1: Reduce science frames of the Q and/or U observations in the polarization P1 mode

sph_zpl_science_p23: Reduce science frames of the Q and/or U observations for the polarization P2 and P3 modes.

sph_zpl_science_imaging: Reduce science frames in the imaging modes.

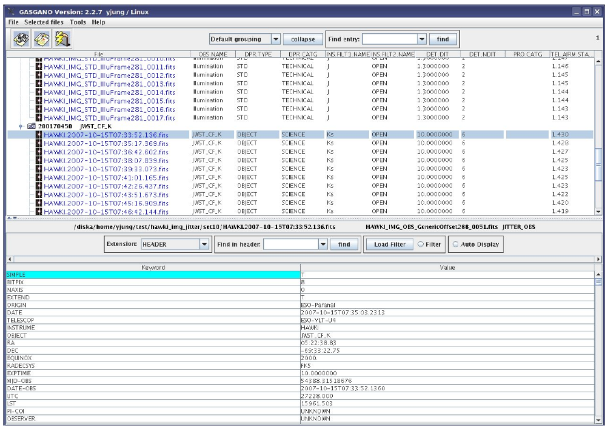


Figure 3.1: The Gasgano main window

3.2 An introduction to Gasgano and EsoRex

Before being able to call pipeline recipes on a set of data, the data must be opportunely classified, and associated with the appropriate calibrations. The Data Classification consists of tasks such as: "What kind of data am I?", e.g. , DARK, "to which group do I belong?", e.g., to a particular Observation Block or template. Data Association is the process of selecting appropriate calibration data for the reduction of a set of raw science frames. Typically, a set of frames can be associated if they share a number of properties, such as instrument and detector configuration. As all the required information is stored in the FITS headers, data association is based on a set of keywords (called "association keywords") and is specific to each type of calibration. The process of data classification and association is known as data organisation. The DO Category is the label assigned to a data type as a result of data classification. An instrument pipeline consists of a set of data processing modules that can be called from different host applications, either from the command line with Esorex, from the automatic data management tools available at Paranal, or from the graphical Gasgano tool. Gasgano is a data management tool that simplifies the data organisation process, offering automatic data classification and making the data association easier (even if automatic association of frames is not yet provided).

Gasgano determines the classification of a file by applying an instrument specific rule, while users must provide this information to the recipes when they are executed manually using Esorex from the command line. In addition, Gasgano allows the user to execute directly the pipeline recipes on a set of selected files.

3.2.1 Using Gasgano

To get familiar with the SPHERE pipeline recipes and their usage, it is advisable to begin with Gasgano, because it provides a complete graphic interface for data browsing, classification and association, and offers several other utilities such as easy access to recipes documentation and preferred data display tools. Gasgano can be started from the Command Line Interface in the following way: gasgano & Figure 4.2.1 shows the Gasgano main window.

With the pull-down-menu File->Add/Remove Files directories containing SPHERE data can be added. The data are hierarchically organised as preferred by the user. After each file name are shown the classification, the template id, the original file name, the template exposure number and the number of exposures in the template. More information about a single frame can be obtained by clicking on its name: the corresponding FITS file header will be displayed on the bottom panel, where specific keywords can be opportunely filtered and searched. Images and tables may be easily

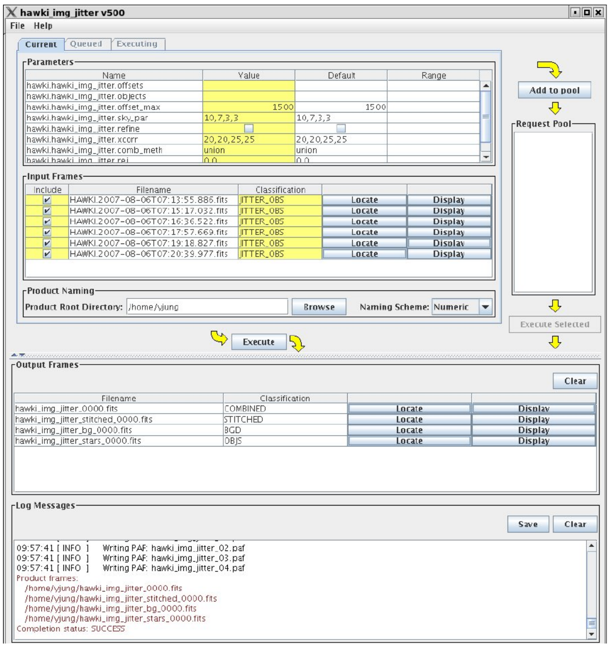


Figure 3.2: The Gasgano recipe execution window

displayed using the viewers specified in the appropriate Preferences fields.

Frames can be selected from the main window with a <CTRL>-left-click for processing by the appropriate recipe: on Figure 4.2 a set of calibration FITS-files have been selected and after selecting the appropriate recipe, the depicted Gasgano recipe execution window will open, having all the specified files listed in its Input Frames panel. Help about the recipe may be obtained from the Help menu. Before launching the recipe, its parameters may be modified on the Parameters panel (on top). The window contents might be saved for later use by selecting the Save Current Settings entry from the File menu, as shown in figure. At this point the recipe can be launched by pressing the Execute button. Messages from the running recipe will appear on the Log Messages panel at bottom, and in case of successful completion the products will be listed on the Output Frames panel, where they can be easily viewed and located back on the Gasgano main window. Please refer to the Gasgano User's Manual [RD9] for a more complete description of the Gasgano interface.

3.2.2 Using EsoRex

EsoRex is a command line utility for running pipeline recipes. It may be embedded by users into data reduction scripts for the automation of processing tasks. On the other side, EsoRex doesn't offer all the facilities available with Gasgano, and the user must classify and associate the data using the information contained in the FITS header keywords (see Section 6). The user should also take care of defining the input set-of-frames and the appropriate configuration parameters for each recipe run: The set-of-frames: Each pipeline recipe is run on a set of input FITS data files. When using EsoRex the file names must be listed together with their DO category in an ASCII file, the set-of-frames (SOF), that is required when launching a recipe. Here is an example of SOF, valid for the sph_ird_instrument_flat recipe:

```
/data/calib/master_dark.fits                  IRD_MASTER_DARK
/data/2011-03-27/raw_flat_bright_DIT_0.fits   IRD_FLAT_FIELD_RAW
/data/2011-03-27/raw_flat_bright_DIT_1.fits   IRD_FLAT_FIELD_RAW
/data/2011-03-27/raw_flat_bright_DIT_2.fits   IRD_FLAT_FIELD_RAW
```

Note that the SPHERE pipeline recipes do not verify the correctness of the DO category specified by the user in the SOF. The reason of this lack of control is that SPHERE recipes are just one component of the complete pipeline running on Paranal, where the task of data classification and association is carried out by separate applications. Using Gasgano as an interface to the pipeline recipes will however ensure a correct classification of all the data frames, assigning the appropriate DO category to each one of them (see section 4.2.1). A recipe handling an incorrect SOF may stop or display unclear error messages at best. In the worst cases, the recipe would apparently run without any problem, producing results that may look reasonable, but are actually flawed.

EsoRex syntax:

The basic syntax to use ESOREX is the following:

```
esorex [esorex_options] recipe_name [recipe_options] set_of_frames
```

To get more information on how to customise ESOREX (see also [13]) run the command:

```
esorex --help
```

To generate a configuration file esorex.rc in the directory $HOME/.esorex run the command:

```
esorex --create-config
```

A list of all available recipes, each with a one-line description, can be obtained using the command:

```
esorex --recipes
```

All recipe parameters (aliases) and their default values can be displayed by the command

```
esorex --params recipe_name
```

To get a brief description of each parameter meaning execute the command:

```
esorex --help recipe_name
```

To get more details about the given recipe give the command at the shell prompt:

```
esorex --man-page recipe_name
```

3.2.2.1 Recipe configuration:

Each pipeline recipe may be assigned an EsoRex configuration file, containing the default values of the parameters related to that recipe. The configuration files are normally generated in the directory $HOME/.esorex, and have the same name as the recipe to which they are related, with the file name extension .rc. For instance, the recipe sph_ifs_master_dark has its EsoRex generated configuration file named sph_ifs_master_dark.rc, and is generated with the command:

```
esorex --create-config sph_ifs_master_dark
```

The definition of one parameter of a recipe may look like this:

```
# --ifs.master_dark.clean_mean.reject_high
# Reject high.
ifs.master_dark.clean_mean.reject_high=2
```

In this example, the parameter ifs.master_dark.clean_mean.reject_high (controlling the number of outliers at the high end to discard when combining frames) is set to the value 2. In the configuration file generated by EsoRex, one or more comment lines are added containing information about the possible values of the parameter, and an alias that could be used as a command line option. The recipes provided by the SPHERE pipeline are designed to be usable in a cascade of data reduction steps, each controlled by its own parameters. For this reason and to prevent parameter name clashes we specify as parameter prefix not only the instrument name but also the name of the step they refer to. Shorter parameter aliases are made available for use on the command line. The command

```
esorex --create-config recipe_name
```

generates a default configuration file recipe_name.rc in the directory $HOME/.esorex. A recipe configuration file different from the default one can be specified on the command line:

```
esorex --recipe-config=my_alternative_recipe_config
```

Recipe parameters are provided in Section 9. More than one configuration file may be maintained for the same recipe but, in order to be used, a configuration file not located under $HOME/.esorex, or having a name different from the recipe name, should be explicitly specified when launching a recipe.

3.2.2.2 Recipe execution:

A recipe can be run by specifying its name to EsoRex, together with the name of a set-of frames. For instance, the following command line would be used to run the recipe sph_ifs_master_dark for processing the files specified in the set-of-frames sph_ifs_master_dark.sof:

```
esorex sph_ifs_master_dark sph_ifs_master_dark.sof
```

The recipe parameters can be modified either by editing directly the used configuration file, or by specifying new parameter values on the command line using the command line options defined for this purpose. Such command line options should be inserted after the recipe name and before the SOF name, and they will supersede the system defaults and/or the configuration file settings. For instance, to set the sph_ifs_master_dark reject_high parameter to 4 the following should be typed:

```
esorex sph_ifs_master_dark --ifs.master_dark.clean_mean.reject_high=4
            sph_ifs_master_dark.sof
```

For more information on EsoRex, see [13].

3.3 Example IFS Reduction

In this basic example for a data reduction for IFS a science product is created from raw input data that was taken in the lab. The steps that are carried out in this reduction is the creation of a master dark, a creation of a master detector flat field, the determination of the spectra regions, a wavelength calibration an creation of a master instrument flat field and finally the reduction if the science images. These steps will most certainly be the ones carried out to reduce the vast majortiy of actual observations.

The example described here will reproduce the test IFS-01 from RD11!

3.3.1 Data set

The data sets for this example IFS reduction can be obtained from:

http://www.mpia-hd.mpg.de/SPHERE/sphere-web/TestData/PAE/ifs_pae_testdata.tar.gz [1]

Upon untarring this, you will find a directory structure based at the directory where you issued the untar command. The .sof file examples below may have to be adapeted to contain this base path!

3.3.2 Creation of the master dark

To create the master dark, the recipe sph_ifs_master_dark must be run. Wherever you choose to run the recipe, create a file called e.g. "master_dark.sof" which should look like this (mind the base directory!):

```
dark/SPHERE_IRDIFS_DARK_IFS129_0003.fits        IFS_DARK_RAW
```

Now run

```
esorex sph_ifs_master_dark master_dark.sof
```

to execute the recipe. By the way: A call with

```
esorex --man-page sph_ifs_master_dark
```

provides you with a help page.

The recipe will run for a less than a minute. It will then output (your mileage may vary!):

```
    ....
[ INFO   ] esorex: [tid=000] Created product master_dark.fits (in place)
[ INFO   ] esorex: [tid=000] Created product static_badpixels.fits (in place)
[ INFO   ] esorex: [tid=000] 2 products created
[ INFO   ] esorex: [tid=000] Recipe operation(s) took
20 seconds to complete.
[ INFO   ] esorex: [tid=000] Size of single raw input frame =    167.88 MB
[ INFO   ] esorex: [tid=000] => processing rate of      8.41 MB/sec
```

The first of these is the master dark, the second the bad pixels in a separate file. View it with your favourite FITS viewer and compare with the input files to verify that it is indeed the mean of the inputs. Notice that the master_dark.fits file has 3 additional extensions – look at all of them using e.g. ds9 by calling (1 stands for the first extension):

```
ds9 master_dark.fits[1]
```

Later in this manual you can find a description of the recipe in detail and what is stored in the other extensions.

[1] You will be asked for a password which you can obtain from mfeldt@mpia.de!

3.3.3 Creating the master detector flat(s)

Similarly to the creation of the master_dark, now create a file called master_dff.sof with the content:

```
oldauxdata/SPHERE_IRDIFS_FLAT_IFS207_0009.fits IFS_DETECTOR_FLAT_FIELD_RAW
oldauxdata/SPHERE_IRDIFS_FLAT_IFS207_0010.fits IFS_DETECTOR_FLAT_FIELD_RAW
master_dark.fits                               IFS_MASTER_DARK
```

Run the recipe using

```
esorex sph_ifs_master_detector_flat master_dff.sof
```

You will now see that the recipe produces several flat fields. The output of this recipe is composed by four files: master_detector_flat.fits; preamp_flat.fits; large_scale_flat.fits and dff_badpixels.fits. The preamp_flat is a flat field that shows just the pre-amplifier component. You can see that within one preamplifier region the values of this flat is always the same. The large scale flat represents a large scale, smooth flat field part. The master_detector_flat.fits shows “usual” pixel-to-pixel flat with the preamplifier part divided out (in this example here, the large scale flat is not divided out. But the master_detector_flat recipe allows also a large scale flat to be given as input which, when provided, is then divided out from the produced master_detector_flat.fits).

The complicated break down of flat fields for IFS is due to the need to achieve great flat field stability while allow for the minimum amout of time needed to take calibration data. The three flat field parts are later used to “reassemble” the flat for the science reduction.

You should also experiment with using different parameters. For example, rerun the recipe with

```
esorex sph_ifs_master_detector_flat \
       --ifs.master_detector_flat.badpix_lowtolerance=0.9 \
       master_dff.sof
```

The detector flat just created was recorded at a wavelength of 1020nm. You should now move the resulting file master_detector_flat.fits to someting called master_dff_1020.fits. After this operation, you must create additional detector flats for 1230nm using a .sof-file structured like this:

```
oldauxdata/SPHERE_IRDIFS_FLAT_IFS207_0013.fits IFS_DETECTOR_FLAT_FIELD_RAW
oldauxdata/SPHERE_IRDIFS_FLAT_IFS207_0014.fits IFS_DETECTOR_FLAT_FIELD_RAW
master_dark.fits                               IFS_MASTER_DARK
```

, one for 1300nm with .sof

```
oldauxdata/SPHERE_IRDIFS_FLAT_IFS207_0017.fits IFS_DETECTOR_FLAT_FIELD_RAW
oldauxdata/SPHERE_IRDIFS_FLAT_IFS207_0017.fits IFS_DETECTOR_FLAT_FIELD_RAW
master_dark.fits                               IFS_MASTER_DARK
```

, and a broad-band white light detector flat using .sof

```
oldauxdata/SPHERE_IRDIFS_FLAT_IFS207_0004.fits IFS_DETECTOR_FLAT_FIELD_RAW
oldauxdata/SPHERE_IRDIFS_FLAT_IFS207_0006.fits IFS_DETECTOR_FLAT_FIELD_RAW
master_dark.fits                               IFS_MASTER_DARK
```

. Of course you can pick any sensible name to move the output to that comes to your mind, but for the sake of being able to blindly use the example .sof files below we suggest master_dff_1230.fits, master_dff_1300.fits, and master_dff_white.fits.

3.3.4 Creating the reference spectra positions file

For IFS, the wavelengths associated which each pixel are stored in a so called "pixel description table" (PDT). This PDT contains information about each pixel in terms of whether it is inside a spectral region, which spectral region/lenslet is associated with it, what its associated wavelength is etc. There is one standard PDT, usually at the "zero" dithering position, that is a reference PDT for higher level recipes. PDT's at other dithering positions are then calculated by the the pipeline. The recipes responsible for creating this reference PDT is the sph_ifs_spectra_positions recipe and also the wavelength calibration recipe. The spectra_positions recipe does not use data to assigne wavelengths, but only uses a model of IFS to create preliminary wavelength association for the wavelength calibration. The spectra_positions main responsibility is to acutally determine the positions of the spectra regions on the detector. This is done by thresholding the input image and using cross-correlation and fitting methods to update the parameters of the IFS lenslet model, describing where spectra fall. To run the recipe on the example data, create a spectra_pos.sof file as here:

```
specpos/SPHERE_IRDIFS_SPECPOS129_0001.fits IFS_SPECPOS_RAW
master_dark.fits                           IFS_MASTER_DARK
```

The recipe will then produce a PDT as a FITS file with 5 extensions. Look at each extension in turn to see what is in them (some extensions will just contain 0s). The first, main, data unit contains the wavelength associatoins. The second contains the spectra id, the third the lenslet IDs, the fourth the wavelength widths and the fifth the second derivative of the wavelength solution. Since the model is linear, this is zero. After the wavelength calibration recipe has been run, it will be filled.

3.3.5 Wavelength calibration

The wavelength calibration can be carried out now. The corresponding wavecal.sof should look like this for the example data:

```
wave/SPHERE_IRDIFS_WAVE129_0001.fits IFS_WAVECALIB_RAW
spectra_positions.fits               IFS_SPECPOS
master_dark.fits                     IFS_MASTER_DARK
```

Calling the sph_ifs_wave_calib recipe will produce a new PDT saved in pdt_wave_calib.fits (also a very long log file is produced...). Compare the FITS file with spectra_positions.fits output from the spectra_positions recipe. There is only very little difference – in the current example data case, many spectra wavelength fits were not within specs, so for these the wavelength association defaulted to the model. Only some spectra have been recalibrated. To see the spectra that have not been fitted well, the 6th extension can be viewed, e.g. ds9 pdt_wave_calib.fits[6]. All the marked spectra have not been fitted well and defaulted to the model.

3.3.6 Creating an IFU flat field

Now that the spectra wavelengths have been calibrated, one can create a flat field for the lenslet itself which does not contain the detector part and is hence dither independent. This flat field is called an IFU flat field and is the first product in the normal cascade presented in the lenslet plane rather than the detector plane. The resulting image is much smaller. The ifu_flat.sof file for the instrument flat looks something like this:

```
ifsflat/SPHERE_IRDIFS_FLAT129_0001.fits    IFS_FLAT_FIELD_RAW
pdt_wave_calib.fits                        IFS_WAVECALIB
spectra_positions.fits                     IFS_SPECPOS
```

```
master_dff_1020.fits          IFS_MASTER_DFF_LONG1
master_dff_1230.fits          IFS_MASTER_DFF_LONG2
master_dff_1300.fits          IFS_MASTER_DFF_LONG3
master_dff_white.fits         IFS_MASTER_DFF_LONGBB
master_dff_white.fits         IFS_MASTER_DFF_SHORT
preamp_flat.fits              IFS_PREAMP_FLAT
master_dark.fits              IFS_MASTER_DARK
```

The recipe to construct the IFU flat is actually the same as the one used for instrument flat above, sph_ifs_instrument_flat. This is because the input raw data is identical, and only the provided calibration files differ: if a IFS_WAVE_CALIB frame is provided an IFU flat is produced, if not, a normal instrument flat is produced.

The files master_dff*.fits are large scale lamp flats. The preamp flat was produced by the master_detector_flat recipe, in order to ensure optimum results you should run the white-light flat fields last (files with this name get overwritten everytime you call the recipe) - or chose another filename for the one to be used here.

Now you are ready to create the IFU flat with:

```
esorex sph_ifs_instrument_flat ifu_flat.sof
```

Look at the resulting ifu_flat.fits file to see what it looks like.

3.3.7 Science reduction

Finally now the science reduction. Without using dithering, the sof file (in this example its filename is science.sof) looks quite simple:

```
sciencej/SPHERE_IRDIFS_OBJECT_IFS129_0001.fits    IFS_SCIENCE_DR_RAW
sciencej/SPHERE_IRDIFS_OBJECT_IFS129_0002.fits    IFS_SCIENCE_DR_RAW
master_dff_1020.fits                              IFS_MASTER_DFF_LONG1
master_dff_1230.fits                              IFS_MASTER_DFF_LONG2
master_dff_1300.fits                              IFS_MASTER_DFF_LONG3
master_dff_white.fits                             IFS_MASTER_DFF_LONGBB
master_dff_white.fits                             IFS_MASTER_DFF_SHORT
pdt_wave_calib.fits                               IFS_WAVECALIB
ifs_ifu_flat.fits                                 IFS_IFU_FLAT_FIELD
static_badpixels.fits                             IFS_STATIC_BADPIXELMAP
master_dark.fits                                  IFS_MASTER_DARK
```

The recipe run in this way

```
esorex sph_ifs_science_dr science.sof
```

Congratulations, you have reduced your first set of IFS data!

3.4 Example IRDIS Reduction

3.4.1 Data set

The data set used in this example is the one used to generate the test sequence IRDIS-01 in RD11. It can be obtained from http://www.mpia.de/SPHERE/sphere-web/IRDIS-01.tar.gz. This countains the raw data files used in the test described in RD11 and outlined below. There is no subdirectory structure in the tar file, so it is best to create a suitable subdirectory from your working directory and unpack the tar file in there. the following examples will assume that this directiry is named "Raw".

3.4.2 Creation of the master dark

To create the master dark, the recipe sph_ird_master_dark must be run. In the IRDIS_EXAMPLE_DATA directory create a file called e.g. "master_dark.sof" which should look like this:

```
Raw/SPHERE_IRDIFS_DARK_IRDIS210_0001.fits                IRD_DARK_RAW
Raw/SPHERE_IRDIFS_DARK_IRDIS210_0002.fits                IRD_DARK_RAW
```

Now run

```
esorex sph_ird_master_dark \
       --ird.master_dark.clean_mean.reject_low=0 \
       --ird.master_dark.clean_mean.reject_high=0 \
       master_dark.sof
```

to execute the recipe. By the way: A call with

```
esorex --man-page sph_ird_master_dark
```

provides you with a help page.

The recipe will run for a less than a minute. It will then output (your mileage may vary):

```
    ....
[ INFO  ] esorex: Created product master_dark.fits (in place)
[ INFO  ] esorex: Created product static_badpixels.fits (in place)
[ INFO  ] esorex: 2 products created
[ INFO  ] esorex: Recipe operation(s) took
43 seconds to complete.
[ INFO  ] esorex: Total size of 3 raw input frames  =   729.95 MB
[ INFO  ] esorex: => processing rate of    16.96 MB/sec
```

The first of these is the master dark, the second the bad pixels in a separate file. View it with your favourite FITS viewer and compare with the input files to verify that it is indeed the mean of the inputs. Notice that the master_dark.fits file has 3 additional extensions – look at all of them using e.g. ds9 by calling (1 stands for the first extension):

```
ds9 sph_ird_master_dark.fits[1]
```

Later in this manual you can find a description of the recipe in detail and what is stored in the other extensions.

3.4.3 Creating the master flat

Similarly to the creation of the master_dark, now create a file called master_flat.sof with the content:

```
Raw/SPHERE_IRDIFS_FLAT_IRDIS210_0004.fits        IRD_FLAT_FIELD_RAW
Raw/SPHERE_IRDIFS_FLAT_IRDIS210_0005.fits        IRD_FLAT_FIELD_RAW
Raw/SPHERE_IRDIFS_FLAT_IRDIS210_0006.fits        IRD_FLAT_FIELD_RAW
Raw/SPHERE_IRDIFS_FLAT_IRDIS210_0007.fits        IRD_FLAT_FIELD_RAW
Raw/SPHERE_IRDIFS_FLAT_IRDIS210_0008.fits        IRD_FLAT_FIELD_RAW
```

Run the recipe using

```
esorex sph_ird_instrument_flat master_flat.sof
```

Again note the the resuling file, irdis_flat.fits has in total 4 data units/extensions.

You shoul also experiment with using different parameters. For example, rerun the recipe with

```
esorex sph_ird_instrument_flat \
       --ird.instrument_flat.threshold=0.9 \
       master_flat.sof
```

look at the output irdis_flat.fits and the first (badpixel) extension irdis_flat.fits[1] to see the difference.

You may also add a line to the .sof file like

```
master_dark.fits   IRD_MASTER_DARK
```

to learn about the influcence of supplying a pre-determined dark.

3.4.4 Telescope sky flat creation

In order to correct for large-scale flat field variations caused in the instrument's optical train, sky flats are used. These can be crated by using e.g. a file called tff.sof with the following contents:

```
Raw/SPHERE_IRDIFS_FLAT_IRDIS178_0003.fits          IRD_TFF_RAW
Raw/SPHERE_IRDIFS_FLAT_IRDIS178_0004.fits          IRD_TFF_RAW
master_dark.fits
IRD_MASTER_DARK
irdis_flat.fits
IRD_FLAT_FIELD
```

Then you call the appropriate recipe with

```
esorex sph_ird_tff tff.sof
```

In a typical science reduction, both flat fields will be sued, the "instrument flat" being attached to the movable detector while the large-scale sky flat remains fixed on the sky coordinate system.

3.4.5 Distortion Map

To correct for the instruments dostortion, a specific map is generated from dedicated calibration data. The corresponding recipe is called "sph_ird_distortion_map", and a good "distort.sof" would look like:

```
Raw/SPHERE_IRDIS072_0004.fits
IRD_DISTORTION_MAP_RAW
master_dark.fits                         IRD_MASTER_DARK
irdis_flat.fits                          IRD_FLAT_FIELD
```

In the test, the call for the recipe was

```
esorex sph_ird_distortion_map \
       --ird.distortion_map.threshold=2.0 \
         distort.sof
```

You can experiment with the threshold parameter to learn why it is important. Like all other recipes that involve CPL functions that do thresholding and peak finding, this one is quite sensitive to the threshold provided and a look at the data may be required before calling the recipe.

Note that since no reference grid is supplied, the recipe assumes that the data provided *is* the reference and generates a reference point table. This is done on the left quadrant. On the right quadrant, which contains of course the identical grid of points, distortion is subsequently measured. You should thus see very small values in the extensions 8 and 12 of the resulting file distortion_map.fits, denoting the distortion in x and y at each pixel.

3.4.6 Star center table creation

Now before actually doing the science reduction it is currently necessary to run a prototype version of the sph_ird_star_center recipe. This recipe is responsible to create a table of field centers, which are crucial for any science reduction with IRDIS. To create this table, first create star_center.sof as

```
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0034.fits      IRD_STAR_CENTER_WAFFLE_RAW
Raw/star_center_maskR.fits                       IRD_STATIC_BADPIXELMAP
master_dark.fits                                 IRD_MASTER_DARK
irdis_flat.fits                                  IRD_FLAT_FIELD
```

and then run

```
esorex sph_ird_star_center \
    --ird.star_center.sigma=1000 \
    --ird.star_center.coll_alg=1 \
    --ird.star_center.nsources=4  \
    star_center.sof
```

which will create a prodcut called star_center.fits that contains a fits table carrying the center coordinates found, a time stamp, and the DMS (Detector Motion Staqge) position during the exposure. Not that as again peak finding and thresholding is involved, the corresponding parameters passed to the recipe are quite sensitive, and experimenting is always welcome!

Note also thet athe star_center_maskR.fits file is not actually raw data, but simply a mask provided along with pipeline! You ar efree to adapt this mask to improve results...

3.4.7 Reducing the science data

As the last step the science data is reduced. There are several different recipes available for IRDIS for this step, depending on tne mode and the desired algorithm. The standard recipe is the DBI recipe, sph_ird_science_dbi. To run this, create a new science_dbi.sof file which has to be identical to the star_center.sof except for the raw file tag names:

```
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0038.fits      IRD_SCIENCE_DBI_RAW
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0039.fits      IRD_SCIENCE_DBI_RAW
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0040.fits      IRD_SCIENCE_DBI_RAW
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0041.fits      IRD_SCIENCE_DBI_RAW
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0042.fits      IRD_SCIENCE_DBI_RAW
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0043.fits      IRD_SCIENCE_DBI_RAW
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0044.fits      IRD_SCIENCE_DBI_RAW
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0045.fits      IRD_SCIENCE_DBI_RAW
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0046.fits      IRD_SCIENCE_DBI_RAW
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0047.fits      IRD_SCIENCE_DBI_RAW
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0048.fits      IRD_SCIENCE_DBI_RAW
```

```
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0049.fits          IRD_SCIENCE_DBI_RAW
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0050.fits          IRD_SCIENCE_DBI_RAW
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0051.fits          IRD_SCIENCE_DBI_RAW
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0052.fits          IRD_SCIENCE_DBI_RAW
Raw/SPHERE_IRDIFS_OBJECT_IRDIS210_0053.fits          IRD_SCIENCE_DBI_RAW
master_dark.fits                                     IRD_MASTER_DARK
irdis_flat.fits                                      IRD_FLAT_FIELD
distortion_map.fits                                  IRD_DISTORTION_MAP
star_center.fits                                     IRD_STAR_CENTER
```

Note that here not all the science frames in the science_science directory are included. This is simply to make sure you dont have to wait for several hours for all the data to be reduced!

```
esorex sph_ird_science_dbi  science_dbi.sof
```

and the recipe writes as main product a file called science_dbi.fits. Congratulations, youre first IRDIS science reductaion has been achieved. Now you can play around with the pipeline to extend your experience and learn about the options you have. The quality of the test data provided is not outstanding, but you can e.g. try to add the option "–ird.science_dbi.use_sdi=TRUE" to the esorex command line (*after* the recipe name, as usual!) to get a spectral difference image between the two channels...

3.5 Example ZIMPOL Reduction

In this basic example of the data reduction for ZIMPOL a calibrated product is created from raw input data that was taken for the "fast polarimetry" mode in the lab. The steps are carried out in this reduction is the creation of a master bias, a creation of a master dark, a creation of a intensity flat field, the determination of the modulation/de-modulation efficiency, and finally creation of the resulting reduced (collapsed) "science image".

3.5.1 Data set

The data sets for this example ZPL reduction can be obtained from:

- http://www.mpia-hd.mpg.de/SPHERE/sphere-web/TestData/PAE/ZIMPOL-01.tar.gz

The archive contains the "fast polarimetry" raw datasets with the pre-defined step-by-step (recipe-by-recipe) data reduction schem. Below we will only describe how to reduce ZIMPOL data for the fast polarimetry mode. Create a subdirectory of your current working directory named e.g. "Raw", change inside and unzip the donloadad tar file. Then go back up one level. You're set to start!

3.5.2 Creating a master bias

As for the other instruments in SPHEREE, you first need to create a .sof file telling the location of your raw data. In this case, it looks extremely simple:

```
Raw/pae3_fp_bias.fits          ZPL_BIAS_RAW
```

Assuming that file is named master_bias.sof, you can then call

```
esorex sph_zpl_master_bias master_bias.sof
```

The recipe will work for some time, mostly busy with preprocessing the complicatedly structured ZIMPOL frames. In the end you will see (your mileage may vary):

```
...
14:13:44 [ INFO ] : [tid=000] Created product zpl_master_bias_cam1.fits (in place)
14:13:44 [ INFO ] : [tid=000] Created product zpl_master_bias_cam2.fits (in place)
14:13:44 [ INFO ] : [tid=000] 2 products created
14:13:44 [ INFO ] : [tid=000] Recipe operation(s) took
26.1 seconds to complete.
14:13:44 [ INFO ] : [tid=000] Size of single raw input frame =    47.46 MB
14:13:44 [ INFO ] : [tid=000] => processing rate of     1.82 MB/sec
```

Note that with ZIMPOL, products are generally created twice - once for each camera. You will notice this when looking at the results, which also appear twice and carry the endings _cam1.fits and _cam2.fits.

3.5.3 Creating a master dark

To create the master dark, the recipe sph_zpl_master_dark must be run. Similar to the master bias, a simple master_dark.sof in this case looks like:

```
Raw/pae3_fp_dark.fits             ZPL_DARK_RAW
zpl_master_bias_cam1.fits         ZPL_MASTER_BIAS_CAM1
zpl_master_bias_cam2.fits         ZPL_MASTER_BIAS_CAM2
```

Again note that we're also feeding the two products created for both cameras to the subsequent recipe. Now run

```
esorex sph_zpl_master_dark master_dark.sof
```

to execute the recipe. The recipe will provide the output:

```
...
[ INFO ] esorex: Created product zpl_master_dark.fits (in place)
[ INFO ] esorex: 1 product created
[ INFO ] esorex: Recipe operation(s) took 1.47 seconds to complete.
[ INFO ] esorex: Size of single raw input frame =    25.17 MB
[ INFO ] esorex: => processing rate of    17.08 MB/sec
```

3.5.4 Calculating the modulation/demodulation efficiency

To compute the modulation/demodulation efficiency using the corresponding raw data, you need to create a "modem.sof" with the following content:

```
Raw/pae3_fp_modem.fits            ZPL_MODEM_EFF_RAW
zpl_master_dark_cam1.fits         ZPL_MASTER_DARK_CAM1
zpl_master_dark_cam2.fits         ZPL_MASTER_DARK_CAM2
zpl_master_bias_cam1.fits         ZPL_MASTER_BIAS_CAM1
zpl_master_bias_cam2.fits         ZPL_MASTER_BIAS_CAM2
```

Run the recipe using

```
esorex sph_zpl_modem_efficiency modem.sof
```

3.5.5 Creating an intensity flat field

Similarly to the creation of the master bias and dark, now create a file called iflat.sof" with the content:

```
Raw/pae3_fp_flat.fits           ZPL_INT_FLAT_FIELD_RAW
zpl_master_dark_cam1.fits       ZPL_MASTER_DARK_CAM1
zpl_master_dark_cam2.fits       ZPL_MASTER_DARK_CAM2
zpl_master_bias_cam1.fits       ZPL_MASTER_BIAS_CAM1
zpl_master_bias_cam2.fits       ZPL_MASTER_BIAS_CAM2
```

There are two algorithms to calculate intensity flat field based either on the collapsing the frames and normalization by the mean value or performing a linear fitting for each single pixel. The parameter "zpl.intensity_flat.collapse=TRUE" acivates the collapsing method.

3.5.6 Science data reduction

reducing the science data is of course the most interesting thing to do. A science.sof should look as follows

```
Raw/pae3_t0.1_fp_star_1.fits              ZPL_SCIENCE_P1_RAW
Raw/pae3_t0.1_fp_star_2.fits              ZPL_SCIENCE_P1_RAW
Raw/pae3_t0.1_fp_star_3.fits              ZPL_SCIENCE_P1_RAW
Raw/pae3_t0.1_fp_star_4.fits              ZPL_SCIENCE_P1_RAW
zpl_master_dark_cam1.fits                 ZPL_MASTER_DARK_CAM1
zpl_master_dark_cam2.fits                 ZPL_MASTER_DARK_CAM2
zpl_master_bias_cam1.fits                 ZPL_MASTER_BIAS_CAM1
zpl_master_bias_cam2.fits                 ZPL_MASTER_BIAS_CAM2
zpl_modem_efficiency_cam1.fits            ZPL_MODEM_EFF_CAM1
zpl_modem_efficiency_cam2.fits            ZPL_MODEM_EFF_CAM2
zpl_master_intensity_flat_cam1.fits       ZPL_INT_FLAT_FIELD_MASTER_CAM1
zpl_master_intensity_flat_cam2.fits       ZPL_INT_FLAT_FIELD_MASTER_CAM2
```

This will apply all the previously created calibration products when calling

```
esorex sph_zpl_science_p1 science.sof
```

Voilá, you have created your first ZIMPOL science output. Note that when operating like this with esorex on the command line, it is the user's responsibility to ensure that the modes (in this case fast polarimetry / P1) match for all the raw files!

Chapter 4

Mathematical Description

Here we describe the mathematical algorithms that are used for data reduction. In addition, this chapter serves as an overview of the general data reduction process.

4.1 Signal propagation through DRH

For a scientific exposure, the most general observation mode for SPHERE, the scientific signal as given by an input flux $S(\alpha, \beta, \lambda)$ results in a detector image, $I(x, y)$ that represents the electrons received by the detector and converted into counts including all instrumental effects. Here the physical (or "sky") coordinates α,β are transformed onto the detector pixels x, y through the dispersive elements, with the mapping

$$S(x, y) = S(x_{\Delta x,\Delta y}(\alpha, \beta, \lambda), y_{\Delta x,\Delta y}(\alpha, \beta, \lambda)),$$

with the pixel to lenslet associations $x_{\Delta x,\Delta y}(\alpha, \beta, \lambda)$ and $y_{\Delta x,\Delta y}(\alpha, \beta, \lambda)$. These pixel to lenslet associations depend on the relative offset between lenslet array and detector, Δx and Δy and are determined during the wavelength calibration procedures. We will henceforth write $S(x, y; \lambda)$ for $S(x_{\Delta x,\Delta y}(\alpha, \beta, \lambda), y_{\Delta x,\Delta y}(\alpha, \beta, \lambda))$, representing the pixelised science image, i.e. a 2-D detector image of the lenslet array that is devoid of instrumental effects. We write the λ dependence here as a reminder that this is a 2D representation of a wavelength cube.

From the entrance into the telescope the scientific signal is affected by several components in an adverse manner, and all these effects have to be removed by the data reduction process in order to achieve maximal scientific output. This is achieved by applying several transformation to the detected image, $I(x, y)$ to reverse the actions of the instrumental and telescope effects. These transformations are in general applied in a sequential manner, reflecting the physical layout of the detecting system, which consists of several components each of which affects the input signal in series. However, it is important to keep this assumption of "sequentially" which underlies most of the principles of astronomical data reduction in mind. In order to allow the removal of the various effects by the instrument/telescope components, one attempts to isolate and measure the effect of each individual component in a calibration procedure which is executed in a separate step to the science observation, either at various times during the observation night, during the preceding day or only at specific times throughout the year.

The data reduction handling for the IFS subsystem of SPHERE provides calibration procedures to measure and correct for the most important instrumental effects. Realizing that the IFS system can essentially be broken down into three relevant parts: the detector including readout electronics, the instrument, including optical components like lenslets and the telescope, including the SPHERE "common path", the signal propagation can be represented by a series of components that act on

Figure 4.1: Schematic representation of the hardware components for IFS from a DRH point of view. The signal is affected by various hardware components which are calibrated out on the data reduction process. For each component the basic mathematical effect is given either as addition, division or multiplication. The first left most effects are all included in the "science" signal $S(x, y; \lambda)$ below.

the input signal $S(x, y; \lambda)$. We show these components schematically in Fig 3.1. Mathematically the signal propagation can be represented with the following equation:

$$I(x, y) = G \times \{DC(x, y) \times \Delta t + B(x, y) + DF(x, y, \lambda) \times IF(x, y, \Delta x, \Delta y, \lambda) \times S(x, y; \lambda)\} + RON, \tag{4.1}$$

where G is the total gain, DC the dark current, B the bias, DF the detector flat response, IF the instrument flat response, RON the readout noise, Δx the detector dither offset in x, Δy the detector offset in y. The exposure time, Δt has the special property (due to the detector technology) that

$$\Delta t = n \times T, \; n \geq 1,$$

where T is a constant exposure time unit, around 1.3sec. Note that $n \geq 1$ and so an exposure time of $\Delta t = 0$ is not possible. This also means that a "bias", defined as the detector response for zero exposure time, can not be measured directly for IFS and IRDIS but has to be inferred.

All the functions for the system components, DC, B, DF, IF and TF are written in detector pixel coordinates, even if the corresponding calibrations may be detector position independent. For example, the instrument flat field is the effect of the lenslet array on the signal, which is a function of lenslet and wavelength but is independent on the detector position. The signal as received in detector pixel coordinates, however, is dependent on the detector offset simply due to a shift in coordinate system. In this sense the functions for the system components defined in the equation above rather represent the detected signal on the detector if all other contributions are zero. The response functions of the detector, the instrument and the telescope are assumed to be linear in the signal S. Linearity of these components, for signals in unsaturated regimes, is part of the SPHERE hardware requirement specification and the linearity assumption is therefore in general justified. An exception are image ghosts (due to optical reflections) and persistence effects.

4.2 Signal propagation reversal

Given the above signal response equation, the inverse can be formulated to infer the original science signal from the detected image:

$$S(x, y; \lambda) = \frac{[I(x, y) - RON] / G - DC(x, y) \times \Delta t - B(x, y)}{DF(x, y, \lambda) \times IF(x, y, \Delta x, \Delta y, \lambda) \times TF(x, y, \Delta x, \Delta y, \lambda)}.$$

The statistical mean of the readout noise should be zero (by choice), simplifying the equation slightly to:

$$S(x, y; \lambda) = \frac{I(x, y)/G - DC(x, y) \times \Delta t - B(x, y)}{DF(x, y, \lambda) \times IF(x, y, \Delta x, \Delta y, \lambda) \times TF(x, y, \Delta x, \Delta y, \lambda)}. \tag{4.2}$$

Knowledge of the functions DC, B, DF, IF and TF and the gain allows one then to determine the scientific signal from the observed detector image. The various functions are determined in the calibration procedures by isolating the relevant components, using a known input source signal S and processing the detector image.

4.3 Signal propagation for bias and dark calibrations

The signal propagation for bias and dark calibration is very simple since the signal S is zero:

$$I(x, y) = G \times [DC(x, y) \times \Delta t + B(x, y)] + RON.$$

Assuming that the statistical mean of the readout noise is zero, the bias and dark term can simply be obtained as

$$DC(x, y) \times \Delta t + B(x, y) = I(x, y)/G.$$

Thus, taking an exposure with closed shutters and dividing by the gain, directly gives the dark+bias contribution. However, note that this depends on the exposure time. Also, since conversion from electrons to counts in the detector also depends on the readout mode, a bias+dark measurement is required for each exposure time and readout mode used in any observation which is to be processed. This is generally true for all detector effects and will be neglected in the further treatment in this chapter (the only consequence is that every measurement is performed for each possible combination of exposure time and readout mode). In the case that a separate measurement of the components DC and B is required, the following description can be used: repeatedly expose the detector for different times Δt thereby obtaining $I(x, y, \Delta t)$ and perform a linear fit to the observed count, $I(x, y, \Delta t) = k(x, y) \times \Delta t + b(x, y)$. Comparison with the above equation directly yields the dark and bias components. Note that this procedure is necessary because an exposure time of 0s is not possible for the infrared detector and so the bias can not be measured using 0s exposures as for optical CCDs.

4.3.1 A special note about the dark calibration and the use of the word "dark" in this document

Even though the calibration plan foresees a master "dark" calibration, and the calibration as well as the result is referred to as "dark calibration" and "dark" or "master dark" throughout this document, this is not really the correct terminology that should be used for this recipe in the case of IRDIS and IFS. Since the dark current is very low for IR detectors, both IRDIS and IFS, what is actually calibrated in this recipe is the so called "Fixed Pattern Noise", or FPN, which represents the spatial variation of the response of pixels to a zero input stimulus. This is dependent on integration time as well as read out mode and may vary on relatively short timescales. To keep with the terminology of the calibration plan we shall continue to refer to this calibration as the "dark" calibration also for the infrared detectors of IFS and IRDIS.

4.4 Signal propagation for the detector flat field

In this case, the detector is illuminated with a uniform lamp of a given wavelength, giving a signal $S(x, y; \lambda) = L(\lambda)$ that is uniform over the detector and depends only on the wavelength, or, more generally, on the spectral energy distribution of the lamp used. Since neither the instrument components (lenslet arrays) or the telescope are involved the detected image is given by:

$$I(x, y) = G \times \{DC(x, y) \times \Delta t + B(x, y) + DF(x, y, \lambda) \times L(\lambda)\} + RON.$$

Knowledge of the bias and dark component from previous measurements, and exploiting the statistical mean of the readout noise of zero gives:

$$DF(x, y, \lambda) = \frac{I(x, y, \lambda)/G - DC(x, y) \times \Delta t - B(x, y)}{L(\lambda)}.$$

This means that the detector flat field response for a given wavelength is measured by taking an exposure of time Δt and subtracting a bias+dark calibration frame with the same exposure time. In general the lamps used for calibration purposes are not perfectly monochromatic, and some detector flats are even taken with a white lamp, giving:

$$DF_L(x,y) = \int [I(x,y,\lambda)/G - DC(x,y) \times \Delta t - B(x,y)]\, L(\lambda)\, d\lambda,$$

where $L(\lambda)$ is the normalized wavelength emission of the calibration lamp L used. Since the actual quantity required in equation 4.2 is $DF(x,y,\lambda)$, it is necessary to extrapolate from a series of $DF_L(x,y)$ for different calibration lamps, L = 1...N. In practice, the calibration lamps used for SPHERE have a small bandwidth and can be assumed to be monochromatic (except for the broad band lamp), giving directly $DF(x,y,\lambda_L)$. Since only a finite number of such calibration lamps are available, it is not possible to determine $DF(x,y,\lambda)$ for every wavelength directly. Rather, determination of $DF_L(x,y)$ for all monochromatic calibration lamps can be used to construct a fit function for every pixel, $f_{x,y}(\lambda)$ which in turn can be used to construct an estimate of $DF(x,y,\lambda)$ for every wavelength. The accuracy of this then depends on the number of monochromatic calibration lamps used, and how well the wavelength dependence of the pixel response can be described by the fitting function chosen.

4.5 Signal propagation for the instrument flat field

4.5.1 Instrument flat field for IFS

For the instrument flat field for the IFS, the set-up is similar to the detector flat field, except that the lenslet array and some related optical components are in the light path. The equation for the signal propagation, eq. 4.1 becomes:

$$I(x,y) = G \times \{DC(x,y) \times \Delta t + B(x,y) + DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda) \times L(\lambda)\},$$

where $L(\lambda)$ is the spectral energy distribution of the calibration lamp and we have assumed that the mean of the readout noise is zero. Now, the calibration measurement is the same as that for the detector flat field, except that we now measure the product $DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda)$ instead of just $DF(x,y,\lambda)$. Since no lenslet array is used in the detector flat exposure, detector flats are independent of the detector position. However, the pixel associated is detector position dependent, and so, in order to measure $IF(x,y,\lambda)$ the pixel positions have to be remapped through the pixel description table before a detector flat is divided out. The main purpose of the sph_ifs_instrument_flat recipe described later is to create a calibration frame which contains only the $IF(x,y,\lambda)$ part and is detector position independent. These IFU flat calibration frames can simply be obtained in any detector position as long as detector flat fields taken at the same detector position are divided out. Again the limited availability of calibration lamps means that the wavelength dependence will be estimated by functional fits, $f_{x,y}(\lambda)$, to a series of measurements at the different calibration wavelengths, such that

$$DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda) = f_{x,y,\Delta x,\Delta y}(\lambda). \tag{4.3}$$

Recipes that need to correct for the instrument flat field use a set of master input detector flat fields in combination with the detector position independent master IFU flat field to construct the function $f_{x,y,\Delta x,\Delta y}(\lambda)$. Which fitting function to use is decided within the recipe and may be a parameter to the recipe plugin. We should note here, that the λ dependence of the lenslet response is expected to be small – and it may in principle be possible to simplify the data reduction process in this case.

For IFS the quantity on the left hand side of equation4.3multiplied by the wavelength association mask as obtained during the wavelength calibration (described further below)

$$SF(x,y) \equiv \int DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda) \times \delta(\lambda - \lambda_{x,y})\, d\lambda \tag{4.4}$$

is referred to as the "Super Flat". In the equation defining SF(x,y) $\delta(\lambda)$ is the Dirac delta function and $\lambda_{x,y}$ is the wavelength associated with the pixel at x,y through the wavelength calibration. This quantity is measured directly in the sph_ifs_instrument_flat recipe, but we reiterate that for reduction of IFS science frames it is not enough to measure the quantity SF(x,y) in one single calibration since the different quantities entering S(x,y) vary on different timescales. Therefore the super fat is reconstructed from separate master calibration files of DF, IF and the wavelength calibration within all science observation and calibration data reductions. Also note that SF(x,y) is dither position dependent.

4.5.2 Instrument flat for IRDIS

Fir IRDIS there is no detector flat field, and there exists only the instrument flat field calibration. Also, since the main observing modes for IRDIS are imaging modes, the wavelength dependence is implicit only and the instrument flat field becomes

$$FF_{imaging}(x,y;F) \equiv \int DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda) \times F(\lambda)\, d\lambda, \tag{4.5}$$

where the flat quantity $F(\lambda)$ is the filter transmission curve. For the DBI mode, the filter will be different for the left and right sub-windows of the detector. Again, note that for IRDIS, as opposed to IFS, the quantity DF and IF are not measured separately, but only $FF(x,y;F)$ is measured. This also means that I is, strictly speaking, dithering dependent.

For the spectroscopy mode, the flat field is defined in an analogous way to IFS as

$$FF_{spec}(x,y;F) \equiv \int DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda) \times F(\lambda) \times \delta(\lambda - \lambda_{x,y})\, d\lambda. \tag{4.6}$$

Again, the flat field is measured in its entirety. It is therefore both filter as well as dithering dependent.

4.6 Finding spectral regions in IFS

For the IFU capabilities of the IFS it is necessary to identify all the regions where spectra fall onto the detector in an automatic way. In principle this needs to be done for every possible detector position in a separate calibration step implemented as the sph_ifs_spectra_positions in the data reduction library. However, the creation of detector position dependent PDTs is done purely in the data reduction recipes: only one "master" PDT table for the standard dithering position is created during calibrations. PDTs for other dithering position are calculated from the dithering position and this master PDT. The simplest algorithm for detecting spectral regions proceeds as follows:

1. Create dark subtracted master calibration frames from the input raw frames. Bad pixels must be flagged/set to zero.
2. Divide this frame by a master detector flat field taken with the broad band (white light) lamp.

3. Apply a threshold algorithm to identify regions with pixel values above a certain threshold value.
4. Assign a label for each connected region.
5. Associate these regions from the regions as expected from a model of the lenslet array. Regions that are either associated to two different lenslet IDs or that have no lenslet ID associated from model are counted and marked.
6. Save label information for each pixel in the pixel description table (PDT). Pixels outside spectral regions are given label 0.

The advantage of this procedure is that it is very simple and the spectral regions have been identified using a clear criterion. This procedure is also very robust to "missing" lenslets or gaps. However, this simple minded approach has the disadvantage of requiring a flat spectra response, that is, the signal along a spectrum must be high and the contrast with regions that do not contain spectra must be high. This is not always the case, since the detector is likely to have a strongly wavelength dependent sensitivity the spectra will not be flat on the detector and in some regions the detector sensitivity may be so low that the contrast is not high enough. In addition this procedure does not take account of the fact that the boundary of spectra do not fall exactly in between pixels; some pixels will be illumination partly by a spectrum, further reducing the contrast with un-illuminated regions. All this means that the performance of this procedure depends rather critically on the choice of the threshold parameter.

An alternative approach, used in the current SPHERE pipeline, uses a model function of the spectra locations to improve on the simple thresholding approach. This model can be provided simply as an image, $M(x', y')$, where

$$M(x', y') = \begin{cases} 0 & off\ spectra \\ 1 & on\ spectra \end{cases},$$

The determination of spectral regions on the observed detector image, $I(x, y)$then just becomes an optimization problem for finding the offset between $\Delta x = x' - x$ and $\Delta y = y' - y$ such that the difference in observed illuminated and predicted spectral regions is minimal and, ideally,

$$I(x + \Delta x, y + \Delta y) = M(x', y').$$

The model itself is derived from the IFS instrument model as described in 6.1. For all the details on the spectra positions procedure please see the recipe description in **??**.

4.7 The IFS wavelength cube and IFS wavelength calibrations

4.7.1 The wavelength cube

For science data reduction purposes of IFU data it is necessary to perform a series of wavelength calibration procedures. Regions on the detector have to be identified where the spectra fall on and every pixel has to be corrected for the wavelength dependent effects. In general, any IFU data at spatial coordinates α, β and at wavelength λ will be constructed from a detector image $I(x, y)$ (obtained with the lenslet array in the optical path) in the following way:

$$IFU(\alpha, \beta, \lambda) = I(x_{\Delta x,\Delta y}(\alpha, \beta, \lambda), y_{\Delta x,\Delta y}(\alpha, \beta, \lambda)),$$

with the pixel to lenslet associations $x_{\Delta x,\Delta y}(\alpha,\beta,\lambda)$ and $y_{\Delta x,\Delta y}(\alpha,\beta,\lambda)$. These pixel to lenslet associations depend on the relative offset between lenslet array and detector and are determined during the spectra positions procedures described in section 3.6. In order to associate the detector pixels with the correct wavelength, a known line spectrum is used to illuminate the detector. Spectral regions are identified and a fit is performed to determine the pixel coordinates of the known line centers of the spectrum. For every lenslet spectrum a table associating the line wavelengths with pixels information is constructed and a fitting/interpolation procedure is used to associate wavelengths for all pixels in between. In this way every pixel will be associated with a lenslet (i.e. α and β coordinates) and a wavelength. Since the procedure makes use of the same instrument set-up as used for the instrument flat procedures the resulting detector image is:

$$I_{\Delta x,\Delta y}(x,y) = G \times \{DC(x,y) \times \Delta t + B(x,y) + DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda) \times S(x,y;\lambda)\}\,.$$

However, contrary to the instrument flat field procedure we are not interested in obtaining $I(x,y,\lambda)$ but rather, we wish to obtain $S(x,y;\lambda)$, the "true" input signal, i.e. the idealized projection of the spectra after having gone through the lenslet array. Using the reversed propagation equation 4.2, we can write

$$S_{\Delta x,\Delta y}(x,y;\lambda) = \frac{I(x,y)/G - DC(x,y) \times \Delta t - B(x,y)}{DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda)}.$$

Thus, a reconstruction of $S(x,y;\lambda)$ can be achieved if the bias and dark current $DC(x,y)\Delta t + B(x,y)$ as well as the instrument flat $DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda)$ are measured accurately. When a white flat field lamp is used as the illumination source the "detector representation" of $S(x,y;\lambda)$ corresponds to the "super flat" S(x,y) defined above.

4.7.2 Wavelength calibration

At the wavelength calibration stage, the information of the pixel to wavelength associations is not yet available (that is rather the result or purpose of the wavelength calibration) the flat fielding has to be performed here using a IFS flat field frame which has not been divided by the detector flat, but is a measure of both detector and IFU flat. That is, DF and IF are not known separately, but rather together. The quantity $DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda)$ integrated over the associated pixel wavelengths is just the super flat field defined in 4.3. So, as a first step, after dark subtraction, the raw wavelength calibration frames are divided by the super flat field (as measured directly in the sph_ifs_instrument_flat recipe).

The flat fielded signal, $S_{\Delta x,\Delta y}(x,y;\lambda)$ is then analysed to detect the spectral lines of the wavelength calibration lamp. The calibration lamp produces very sharp, monochromatic lines. The line profile as observed on the detector are a convolution of the intrinsic line profile, negligible for the calibration lamp lines used, and the instrumental profile (spectrograph resolution). Therefore, the expected line width for these new calibration hardware will be entirely given by the spectrograph resolution (about 2 pixels). Since the calibration lines are sharp, there is a possibility of some additional faint lines due to fringing, and so the positions on the spectra lines have to be determined by pre-selecting the regions close to the expected positions of the lines to avoid the fitting to be performed on some of these fringes (present as local maxima). Alternatively, the data extracted from the spectra region is first passed through a low pass filter to smooth out fringes before fitting is performed – before de-convolving again to assure that the measured FWHM is not affected. The line fringing is not expected to be an important effect for IFS, where spectra resolution is low, but for IRDIS MRS spectroscopy line fringing has to be taken into account. Only the first, simplest method is currently implemented. The peak position for each line is determined by calculating the weighted mean position for a window of a few pixels size around the expected line position. The expected line position is taken from an input model of the spectra positions and the dispersion. Since this input can also be another wavelength calibration product, the wavelength calibration can in principle be performed iteratively.

Once the line positions have been identified, a polynomial of degree $P > 0$ is used to fit the curve of known line wavelengths to measured pixel coordinates. This fit is then used to fill all pixels not covered by lines with wavelength information. If $P > 1$ the second derivative is used to estimate the dispersion. [1]

4.8 Further spectral and flux calibrations

The IFS uses IFU capabilities to create a wavelength data cube as the main science product of every observation. This wavelength data cube needs to be, as much as possible, free of instrumental effects and measure as accurately as feasible the true spectrum of the source. However, the observed spectrum is, just like the detector image, affected by several unwanted effects: the atmosphere as well as telescope subsystem introduce wavelength dependent effects. The observed spectrum is given by:

$$F_{obs}(\lambda; z, r, \theta) = F_{real}(\lambda) \times A_{atm}(\lambda; z) \times A_{corono}(\lambda; r, \theta) \times A_{tel}(\lambda; r, \theta) + T_{atm}(\lambda; z), \qquad (4.7)$$

where $A_{atm}(\lambda; z)$,$A_{chorono}(\lambda; r, \theta)$ and $A_{tel}(\lambda; r, \theta)$ are the attenuation effects of the atmosphere, the coronagraph and the telescope, respectively, $T_{atm}(\lambda; z)$ is the atmospheric transmission at wavelength λ and $F_{real}(\lambda)$ is the true scientific signal to detect, which needs to be reconstructed in the calibration procedure. In order to be able to do this, without the need to obtain calibration frames for every science exposure, it is necessary to model the various instrumental and atmospheric effects individually and measure the relative contributions and model parameters at regular intervals. To this end, the various effects can be disentangles making use of the different dependencies: the atmospheric effects depend on airmass, z but are independent of source location on the detector, whereas telescope and coronagraph affect the signal dependent on the source position within the frame but are independent of air mass. The various components are modeled and calibrated as follows.

4.8.1 Atmospheric absorption

In order to remove the atmospheric dependence, it is necessary to use a spectral model of a known observed source. To avoid a strong dependence of the data reduction pipeline on model dependent quantities, the recipes for atmospheric calibration currently produce simple reduced science frames. These have to be processed further by e.g. dividing by the known star spectra to obtain the atmospheric contribution.

4.8.2 Coronagraph effects

In general, the contribution of the coronagraph is not removed. Frames are processed without division by the coronagraph attenuation to reduce the impact that the incorrect removal of its effect may have on the data quality.

In the rare cases where a removal of the coronagraph effect is explicitly required, exposures can be taken with and without coronagraph and the ratio of the resulting spectra gives directly the coronagraphic effect (except for an unknown normalization constant). However, the fact that measurements are necessary at different points in the field makes this rather problematic if no

[1] Alternatively, a dispersion model which gives the dispersion as a function of wavelength, $D(\lambda)$ is used to create a "guess" pattern of line positions for each lenslet, and this pattern is matched to the observed line pattern (allowing for positional shifts). The "goodness" of fit of the pattern is calculated for each lenslet and can be used to monitor the dispersion stability of the lenslets. The dispersion $D(\lambda)$ model can also be created as an output of the wavelength calibration when the polynomial fit is used to obtain line positions; the wavelength calibration can in fact be regarded as a measurement of the dispersion for each lenslet.

good model of the coronagraphic effect with few parameters is found. It should be possible to determine a functional model, $A_{corono}(\lambda, r, \theta; \alpha_0, \alpha_1, ..., \alpha_N)$ with N parameters $\alpha_0, ..., \alpha_N$, and N being a small number. The parameters of the model and verification will have to be determined in the first weeks after or during commissioning.

4.8.3 Instrumental background and sky background

In 4.1 no additive effects are included except those arising from the detector. However, there are signal contributions from both the sky and the instrument itself which need to be removed in order to obtain the true science signal. Since these are additive effects, they need to be subtracted out after the reverse propagation equation above, 4.2 has been applied. The total signal is

$$S_{tot}(x, y; \lambda) = S_{sky}(x, y; \lambda) + S_{ins}(x, y; \lambda) + S_{sci}(x, y; \lambda),$$

where S_{sky} is the sky background, S_{ins} is the instrument background and S_{sci} is the actual science signal. Note that these signals are additive – and so the calibration/removal of the sky and instrument background follows a similar procedure to the dark calibration, in the sense that they are subtracted from the input frames. Also note that all contributions have a wavelength dependence.

For IFS the contributions of the sky and instrument background are expected to be small and unimportant for the main science objective: the detection and imaging of planets. It is only in special cases, for example when extended source are observed, that the sky and instrument background may significantly affect the science goal of the observations. Therefore, even though recipes are included in the pipeline to calibrate these effects, their applicability will be limited.

4.8.4 Flux normalization calibration

Since the above calibrations are all performed using ratios, it is not possible to determine in this way the absolute flux. In order to do this, one needs to observe a known source and compare total received flux (i.e. detector counts) with the known flux of the source. This requires a separate calibration procedure: sph_ifs_std_phot and sph_ird_ins_throughput. For more details, see the description of the recipes.

4.9 Time dependency impact of systems

In the above treatment of the signal propagation for the individual calibrations, we have not discussed the effect of time variation in the various detector, instrument and telescope systems. None of these systems are perfectly stable in time, meaning that it is necessary to repeat calibration procedures at time intervals which are less than the stability time-scale for the required accuracy. For example, the detector flat field is stable to within 0.1% only for about one hour. This means that the recipes that need to correct for effects that involve the $DF(x, y, \lambda)$ term need to use calibration measurements of this quantity that are maximally one hour old. An alternative in such cases it to use monitoring measurements to construct a model of the time behaviour of the relevant subsystems. In the following table, we list approximate dependencies of the calibration terms:

Term	Variable dependence	Time-scale	Variation
$DC(x, y)$	x, y	1 day	1%
$B(x, y)$	x, y	1 day	1%
$DF(x, y, \lambda)$	x, y	30 mins	0.1%
$DF(x, y, \lambda)$	λ	1 week	0.1%
$IF(x, y, \lambda)$	x, y	1 day	0.1%
$IF(x, y, \lambda)$	λ	1 month	0.1%
$TF(x, y, \lambda)$	x, y	1 week	1%
$TF(x, y, \lambda)$	λ	1 month	1%

It is the responsibility of the observer to make sure that the frames used for the calibrations have the required "freshness" – the pipelines will make no checks for that. Also note that some calibration procedures measure quantities that are combinations of terms with different time variability. In these cases, the acceptable time-scale is given by the smallest acceptable time-scale of the subsystems involved. For example, the instrument flat field procedure measures the term $DF(x, y, \lambda) \times IF(x.y, \lambda)$ which has an acceptable time scale for stability of about 30 mins - 1 hour. In some cases, it is also possible to model the time variability in such a way that only part of the procedure has to be performed in frequent intervals. For example, to perform accurate removal of the detector flat field in wavelength calibration it is in principle necessary to have detector flat frames for all 4 calibration lamps that are all newer than 1 hour. However, modeling the behaviour of the detector flat field as

$$DF(x, y, \lambda) = DF(x, y) \times f(\lambda),$$

and noting that the detector response is very stable in terms of the wavelength dependence, $f(\lambda)$ is almost constant over time, it is possible to only perform measurements of $D(x, y)$ frequently, which requires taking calibration data with only a single lamp. This is the approach taken in SPHERE.

4.10 Clean Mean Algorithm: Basic Frame Combination with outlier rejection

Please note the the spatial derivative pixel rejection described below has not been implemented in SPHERE.

The clean mean algorithm is used to average frames taking into account the possibility of bad pixels and outliers in individual frames. The quality of the detector linearity will therefore be an important contributor to the quality of the data reduction process in SPHERE.. The goal is to achieve an optimal mean frame that is not affected by individual bad or outlying pixels at the same time as keeping the maximum amount of information.

We use iterative clean mean with sigma computation (following that described in ESO's SINFONI Pipeline User Manual) as our baseline frame combination method. For this process, the user sets minimum and maximum allowable intensity values. For a stack of frames, values inside this range are then used to determine an intensity mean and standard deviation, for each pixel position. Pixels with values differing from the mean by $k * std$ are removed. This process is re-iterated n times to generate a final mean-combined image.

We wrote an alternative frame combination script with the aim of achieving superior outlier rejection, compared to the clean mean method. This alternative procedure (presented in [RD2]) uses an iterative median/mean combination outlier rejection strategy that takes advantage of the spatial derivative of an image to better deal with variations from a changing PSF shape or small (sub-pixel) pointing errors. This is particularly important in regions where the PSF slope is steepest: there, a small change in pointing or PSF shape could lead to pixel values being wrongly interpreted as outlier pixels. The spatial derivative method should be effective at dealing with such phenomena, as demonstrated in data reduction for Spitzer IRAC (see [RD3]) and HST ([RD4]).

The code operates by first conducting a biased-median-combination of input frames to create a *best estimate image*. "Biased median" refers to taking the value b positions below the median value, to deal with non-symmetrical noise sources like cosmic ray effects. From this *best estimate image, BEI,* a *spatial derivative array, SDA*, may be calculated using the following equation.

$$SDA(x,y) = maxabs(BEI(x,y)\breve{}[BEI(x-1,y), BEI(x+1,y), BEI(x,y+1), BEI(x,y-1)]) \tag{4.8}$$

Going back to the original input frames, we remove any pixel that differs in value from the corresponding *best estimate image* pixel by more than k times the corresponding pixel value in the SDA. In other words, we reject original input frame pixels that meet the following condition.

$$|original - frame(x,y)\breve{}BEI(x,y)| > k \times SDA(x,y) \tag{4.9}$$

The now-corrected input images are then mean-combined to generate the final image.

4.11 Detector pixel linearity

This algorithm is used for example in the sph_ifs_master_detector_flat recipe to determine the detector linearity for each pixel.

The linearity measurement is used in all recipes where the detector response is assumed to be linear as part of the algorithm. This is the case for all recipes that divide out the detector flat field for example, since the exact detector response is a function of input signal, and extrapolation to the actual input signal from the available detector flat calibration frames is needed (the value $DF(x,y,\lambda)$ in the equation 4.1 is just this linear detector coefficient).

Recipes that need to correct for the detector flat field actually use the detector pixel linearity for the flat fielding, since this gives the correction as a function of detector mean, rather than exposure time. The detector flat to correct for is a function of signal rather than integration time.

As part of this algorithm pixels are identified that do not conform with linearity requirements. Such pixels can also be regarded as "bad pixels" in the sense that their behaviour does not follow the expected behaviour – depending on the required accuracy such pixels may need to be excluded from a frame combination procedure. The identification of bad pixels using this method is possible both for dynamic and static bad pixel identification – but its main use is to identify static bad pixels. The algorithm itself only calculates the reduced χ-square of linear fits to each pixels response and the linear coefficients. Recipes can then use this information to flag certain pixels as bad.

This method to identify bad pixels can also be used to check that the dynamic bad pixel identification works as required and that the dynamic bad pixel identification routine are not affecting further data calibration and reductions adversely.

To determine the detector pixel linearity for monitoring is responsibility part of the sph_ifs_gain recipe, where the response of every pixel to different signal levels is measured and a fit performed.

4.12 Bad Pixel Identification

Bad pixel identification in SPHERE happens on several levels and in several ways. First, there is a distinction between dynamic and static bad pixels. Static bad pixels are due to a property of the detector and are, as the name implies, unlikely to change with time. It may happen that a pixel changes its status from "good" to "bad" in terms of static bad pixel detection (i.e. the pixel breaks), but the converse should in general not occur. Dynamic bad pixels however vary from exposure to exposure. Identifying these is therefore responsibility for all algorithms that combine a set of raw frames into a smaller set of output frames, e.g. the CLEAN MEAN algorithm described in 4.10.

4.12.1 Static Bad Pixel Identification

The identification of static bad (dead or hot) pixels in the data reduction occurs usually during the master dark calibration recipe. When the master dark frames are created, a static bad pixel map is created in the following way:

1. The master dark calibration frames are created from the input frames. As described in section 7, a master dark is created for every exposure time and readout.
2. The individual frames are combined using the clean mean method, effectively removing temporary bad pixels that appear in only a few of the frames.
3. A threshold clipping is applied to the combined frames
4. A smoothed version is subtracted (this is an optional step)
5. A two-pass sigma clipping is applied

This routine ensures that static bad pixels are truly static and are not random chance events. However, note that a reliable identification of truly static pixels requires that the sigma threshold in identifying the bad pixels in each master calibration frame is chosen adequately and that there is a reasonable (i.e. more than about 3) number of exposure time set-ups in the input frames.

A second procedure to detect static bad pixels uses the detector linearity behaviour to determine bad pixels. During the sph_ifs_detector_flat fielding recipe, a detector linearity map is created (see 4.11). This map gives, for every pixel, a measure of the linearity (goodness of fit for a linear fit) and the linearity coefficient. This map can then be used to flag pixels as bad. Many recipes allow a parameter to control the threshold on the linearity to accept/reject pixels. The detector flat recipe itself creates a static bad pixel map in this way, which is the standard bad pixel map input for other recipes.

The static (or "hot") bad pixel map identified using the first method, in the sph_ifs_master_dark recipe, is used primarily for monitoring purposes and to validate the static bad pixels identified in the sph_ifs_master_detector_flat field recipe. For that purpose the detector flat field recipe outputs a quality control parameter that measures the number of pixels that have been identified in one static bad pixel map, but not the other. A large number here usually means that the detector linearity performance has degraded.

4.12.2 Dynamic Bad Pixel Identification

Apart from static bad pixels due to faults in the detector, there are also dynamic bad pixels that are created by transient effects: most notably by cosmic rays. These are identified whenever raw frames are combined. Each frame combination routine, like the clean mean algorithm above, needs to reject pixels that are deemed as bad – in the case of the clean mean algorithm due to their value away from the mean value. Since bad pixels due to cosmic rays can affect neighboring pixels as well as the same pixel at a subsequent readout, all pixels around a bad pixel (in a cross shape) should also be flagged as bad as well in the subsequent frame. This happens, for pixels that are sufficiently outlying, automatically in the clean mean algorithm.

4.12.3 Badpixel treatment in the IRDIS and ZIMPOL science recipes

The science reduction recipes for IRDIS and ZIMPOL use the bad pixel maps from the dark and the flat field recipes to set the badpixels on the resulting reduced science images. The badpixels set in the science frames is always the union of all static bad pixel maps. Both IRDIS and

ZIMPOL use geometric transformations as part of the usual science data reduction process (e.g. for differential imaging or to process dithered frames) and the badpixel maps are transformed along with the image. The algorithm for transforming the badpixel maps is purely geometric in any case even when the image data is transformed using an FFT. Due to serious artefacts that would arise if an FFT is performed on data containing many discontinuities arising from bad pixels, the science recipes interpolate the bad pixels before any transformation on the image data. The (also transformed) bad pixel map however is maintained and used to calculate a final badpixel map on the fully reduced and combined science image that then shows as bad all pixels that had no valid input pixel information.

4.13 Field Center

For all SPHERE instruments accurate determination of the field center is important. The required accuracy currently is 3mas with a goal of 1mas. This is true in particular for all pupil stabilized (or fixed de-rotator) modes. Here frame combination as described in 3.14 requires de-rotation of raw frames and for this the rotation center needs to be determined accurately. The situation as given from the hardware is as follows:

- the DTTS loop and reference slopes calibration ensures that:
  - this center of rotation is also the photo center and
  - this is also the location of coronagraph (all of this with satisfactory accuracy <0.5 mas)
- the DTTS calibration (CPI-TEC-01) outputs the position on the IRDIS detector of the coronagraph (for its internal use)

Currently there are several different field center calibration strategies considered:

1. Calibration within science data reduction recipes: here one uses the science raw frames themselves and the fact that the coronagraph center can be easily determined by determining the center of the region masked by the coronagraph. The AO (DTTS loop) then ensures that this coronagraph center is also the rotation center. For frames taken without coronagraph the star center itself, which is easily determined by finding the peak and Gaussian fitting.
2. Calibration of the center in a dedicated recipe but using the input raw science observation frames. For IRDIS the star center calibration which uses 4 secondary diffraction peaks to extrapolate the center of the star and hence the rotation center.
3. Dedicated calibration recipe using an artificial source. An artificial illumination source is used to perform a dedicated calibration which determines the center of the coronagraph for all possible instrument set-ups. This would be a daytime calibration with a as yet to be determined frequency. A dedicated recipe would reduce these calibration images and return a reference field center to use for frame combination in science data reduction recipes.

Currently, option 2 has been implemented for IRDIS in the sph_ird_star_center recipe. Which of these will be implemented for IFS is still to be decided.

4.14 Frame combination: de-shifting and de-rotating

Frame combination is one of the most crucial steps in the SPHERE data reduction since accurate frame combination including de-shifting and de-rotation allows use of ADI, SDI and other more advanced planet finding algorithms.

In SPHERE the frame combination including de-rotation, scaling (for SDI) and de-shifting is intended to be an integral part of the pipeline on the other hand and to be flexible and modular on the other. This is realised by allowing the choice of frame-combination to be given as an input parameter to recipes, along with all relevant parameters needed for the frame combination algorithm. Currently only a simple ADI and SDI is implemented.

For IRDIS the algorithm of choice to de-shift, de-rotate and de-scale is the FFT. The exact FFT implementation is a separate module and de-coupled from the actual frame-combination code so it can be replaced with different implementation easily. Currently the FFT provided by FFTW is used. The GSL fft routines are also available using a switch (recompilation is necessary).

The FFT rotation routine is augmented by a filter to remove high frequency noise. This filter is a simple top-hat filter which removes all frequencies that have a k-value in the Fourier domain that is above a percentage F of the maximum k-value.

For IFS, the algorithm to de-rotate the individual monochromatic images uses the hexagonal lenslet array geometry and the algorithm GIMROS. GIMROS calculates overlaps between polygons to interpolate the lenslet image onto a rotated grid of hexagons.

4.14.1 GIMROS - Generic IMage ROtation and Scaling

The GIMROS algorithm is specifically created to interpolate the image from a hexagonal grid onto a translated and/or rotated second hexagonal grid.

4.14.1.1 The concept behind GIMROS

The "G" in GIMROS means Generic and indicates that the concept of rotating and scaling image data is somewhat generalized in this algorithm. Not as strictly tied to astronomical purposes like in the IPAC Montage package, where every pixel is projected onto the sky before being mapped onto a new pixel grid, but more general in the sense of allowing more pixel shapes. Of particular interest in SPHERE's context are of course the hexagonal pixels of IFS, but in principle GIMROS allows for all convex polygons as image base elements. In fact, an image representation suitable for GIMROS must be more complex than simply a matrix full of values plus a header that describes how the indices's of matrix elements are related to image co-ordinates.

4.14.1.2 Transforming an image with GIMROS

The primary transformation technique applied by GIMROS will be to map the input fluxes collected on the input pixel grid to a given output pixel grid. The calculation the coordinates of the output pixel grid is determined by the transformation and usually not performed by GIMROS itself. A few helper routines may be devised that perform simple special transforms such as arbitrary rotation about an arbitrary centre, arbitrary shifts in arbitrary directions, and image scaling.

The flux value of any output pixel F(i) will be calculated as the sum over all overlapping input pixels flux values G(j) weighted with the overlap area:

$$F(i) = \frac{\sum G(j) a_{ij}}{\sum a_{ij}}$$

It is obvious that this technique is flux conserving as long as both images are completely and continuously covered with non-overlapping pixels. Gaps between pixels, particularly in the output image, may of course lead to losses of flux.

The overall algorithm of GIMROS then looks as follows:

```
for all output pixels:
```

```
   area_weight = 0
   flux = 0
 find all potentially overlapping input pixels
   for current_input_pixel
   in potentially_overlapping_input_pixels:
  area = overlap_area(current_input_pixel, current_output_pixel)*
       current_input_pixel.weight area_weight += area flux +=
       current_input_pixel.flux * area
  if flux != 0:
   flux /= area_weight
```

The introduction of the current_input_pixel.weight allows for handling bad pixel maps on the input side

The finding of the potentially overlapping input pixels can be done in the following way:

```
1. calculate all centre points of all (transformed) input and output polygons
2. find the longest distance between centres and any edge point throughout the input
3. sort the pixel polygons by the distance of their centre point to the centre point
4. the potentially overlapping pixels are the ones where the distance is below 2*dmax
```

The calculation of the overlap are is done by clipping the (transformed) input polygon to the output polygon and calculating the area of the resulting, clipped polygon. For details of this procedure, see http://www.mpia.de/SPHERE/WIKI/pmwiki.php?n=DRH.FrameCombination.

4.15 Creating wavelength cubes for IFS

One of the last reduction steps in producing calibrated science frames for IFS is to construct a wavelength data cube, $S(\alpha, \beta, \lambda)$. As described before, this is achieved using pixel-to-lenslet association tables. The interpretation of the resulting cubic structure is, however, not straightforward unless some additional geometric transformations are used. The reason is that the lenslet array has a hexagonal rather than rectangular structure. This means that every spatial x, y position in the wavelength cube has a non-trivial associated sky position α,β. How this geometric transformation is performed depends on the level of science reduction. For the basic reduction there is two possible outputs: a result on a hexagonal grid as a FITS table, representing a hexagonal "cube" and a result on a (square) pixel cube. In each case, the way that the interpolation onto the output grid is performed depends on the observing mode: field or pupil-stabilized. We describe the general algorithm for creation of wavelength cube below.

The starting point for cube creation is in any case a series of detector images. Spectra have been identified using the sph_ifs_spectra_locations and sph_ifs_wave_calib routines, which have created a master pixel description table, describing pixel wavelength associations for the standard zero-point dithering position. For each dithering position used in the observations to combine, a new PDT has to be constructed. This is done purely in a software manner – calculating a new PDT for the offset position from the master PDT using the sph_pixel_description_table_new_shift function. These PDTs can then be used to extract a spectrum for every raw input frame – thus providing dither independent information. The extracted spectra are saved in a structure called a "lenslet desciption table" (LDT) which describes the data in terms of the the lenslet "view" of IFS. The LDT is strongly linked to the IFS instrument model, described in 6.1. Extracting the spectra and moving to the lenslet view removes any dithering dependence and spectra can now be combined. This is done on a spectrum by spectrum basis. Once a final combined set of spectra has been crated (one spectrum for each lenslet) the result is saved as a wavalength cube. For this, the hexagonal structure is interpolated onto a hexagonal grid. We have currently implemented a method based on GIMROS to project hexagons onto quadratic pixels.

4.16 Distortion map

The distortion map needs to be measured for IFS and IRDIS. For ZIMPOL no distortion map measurement is currently foreseen, but it may be added at a later stage. For IRDIS the distortion map measures the distortion of the actual detector, $\Delta(x, y) = [\Delta_x(x, y), \Delta_y(x, y)]$, defined by

$$S(x, y) = S'(x + \Delta_x(x, y), y + \Delta_y(x, y)) \tag{4.10}$$

where S(x,y) and S'(x,y) represent the distortion corrected and uncorrected signal respectively. For IFS the distortion map of the lenslet array itself is the relevant quantity, which is defined in an analogous way.

In both cases, the distortion is measured using a grid of artificial sources with known positions. The grid positions is used as an input to the distortion map recipe which detects the actual observed sources and measures their displacement with respect to the expected positions to obtain a vector map. The x and y component of this vector map is fit using a 2D polynomial, giving a smooth representation of the distortion map.

4.17 Astrometry and plate scale solution

The field of view for all SPHERE instruments is very small and astrometry for SPHERE is a two parameter problem: the rotation angle relative to the north direction and the pixel scale. The field center itself is determined in a separate recipe.

To allow the determination of the two parameters, a binary system needs to be observed. The relevant recipes reduced the raw observation frames and automatically detect the central star and the companion. Together with the user input parameters of angle and separation the angle to north and plate scale are derived.

4.18 ZIMPOL measurements

4.18.1 Definitions of terms used for ZIMPOL measurements

In this document and in the SPHERE ZIMPOL calibration plan (RD1), several expressions are used for describing specific parts of a ZIMPOL measurement. The different expressions aim to better distinguish between the entities they describe and are chosen according to ESO definitions described in AD1.

Exposure: In general, an *exposure* is the entity of one or more (NDIT) integrations (*frames*), followed by the readout and storage of the NDIT *frames*.

- For ZIMPOL polarimetric modes (P1, P2, P3), an *exposure* is always the entity of two or multiples of two ($2 \cdot$ NDIT) *frames* (due to the double-phase mode) each followed by detector readout and storage of the $2 \cdot$ NDIT *frames*. A ZIMPOL measurement of one Stokes parameter requires a minimum (NDIT$=$ 1) of one *exposure*, i.e. one *exposure* contains two *frames* or four *sub-frames*.
- For ZIMPOL imaging mode (I1), an exposure means the entity of one or more (NDIT) integrations (*frames*), each followed by detector readout and storage of the NDIT *frames*.

Frame: A *frame* is a single integration and readout of the data acquired during DIT seconds. Two *frames* in sequence in double-phase mode form the minimum of one ZIMPOL *exposure*. Each *frame* has a 'number' $k = 1 \ldots 2 \cdot \mathrm{NDIT}$.

Sub-frame: In general, a *sub-frame* is a part of a *frame*. For ZIMPOL, each *frame* consists of exactly two *sub-frames* independent of observing mode: The image data stored in all odd-numbered, exposed software pixel rows (*sub-frame* i^{A}), and that stored in all even-numbered pixel rows covered by the opaque stripe mask (*sub-frame* i^{B}). In the polarimetric modes P1, P2, P3, the demodulation fills the two *sub-frames* with the two complementary polarization images. The spatial field information in both *sub-frames* is the same, since they have been recorded through the same microlenses and exposed pixels. In imaging mode I1, where no demodulation takes place, the intensity image is stored only in the *sub-frame* i^{A} while the other *sub-frame* i^{B} remains empty.

4.18.2 Description of a ZIMPOL measurement in double-phase mode

To better understand and distinguish the different meanings of *exposure*, *frame* and *sub-frame*, a short explanation is given which is valid for all ZIMPOL double-phase mode measurements. Each frame (number $k = 1...2{\cdot}\mathrm{NDIT}$) contains two sub-frames, i_k^{A} and i_k^{B}. Polarization modulation of the incoming light in combination with the demodulation performed on the CCD sensor during the integration time ensure that these two sub-frames represent an intensity image of two opposite Stokes polarization components (e.g. Q_+ stored in i_k^{A}, Q_- in i_k^{B}). In double-phase mode, the demodulation phase is shifted by half a cycle between each consecutive pair of frames, effectively exchanging the assignments (Q_+ stored in i_{k+1}^{B}, Q_- in i_{k+1}^{A}). In Figure 4.2 a graphical explanation is given.

When a cycle of NDIT frames is taken, the half-wave plate HWP2 is rotated by 45° ("HWP2 flip"), and the cycle of NDIT frames is repeated for the second HWP2 position. When the observations for both HWP2 positions are finished, an optional dithering is performed over NDITHER positions.

4.18.3 ZIMPOL CCD

The two chosen ZIMPOL CCDs are e2v 44-82 bi, in frame transfer mode (one CCD for each of the two cameras). One CCD has 2k×4k pixels (hardware pixels of 15×15 μm size). The half of the CCD (2k×2k) is covered by an opaque mask and is used as buffer storage only, the other half (2k×2k) is exposed to light. The exposed part of the CCD is furthermore equipped with an opaque stripe mask which alternately covers two rows of the CCD and leaves the next two rows open (e.g. row 1 and 2 are covered, 3 and 4 are open, 5 and 6 covered, 7 and 8 open, etc.). An on-chip (TBC) 2×2 binning will reduce this to 1k×1k software pixels (30×30 μm size). Thus, from each camera 1k×1k pixels are effectively read out.

The f/ number at the detector is f/221, leading to an image scale of $\approx 0.117\,\mathrm{mas}/\mu\mathrm{m}$ a pixel scale of 3.5 mas/pixel (for 1k×1k pixels).

The image scale in the sub-frames (1k×0.5k) is doubled in one field direction. To make the image scale symmetric in both field directions, the images will be binned again during the data reduction to 0.5k×0.5k final pixels of size 60×60 μm corresponding to an image scale of$\approx 0.233\,\mathrm{mas}/\mu\mathrm{m}$ or a pixel scale of 7 mas/pixel. This corresponds to about half a resolution element at 600 nm, which is $\lambda/D \approx 15.5\,\mathrm{mas}$ (factor 1.1 oversampling compared to Nyquist criterion at 600 nm).

The total field of view at the ZIMPOL focal plane is about 8" diameter, whereas the entire detector covers only about 3.5×3.5".

An extract of one ZIMPOL CCD can be seen in Figure 4.18.3. There, the boxes with the smallest sizes correspond to the 15 μm sized hardware pixels. The always present on-chip (TBC) 2×2

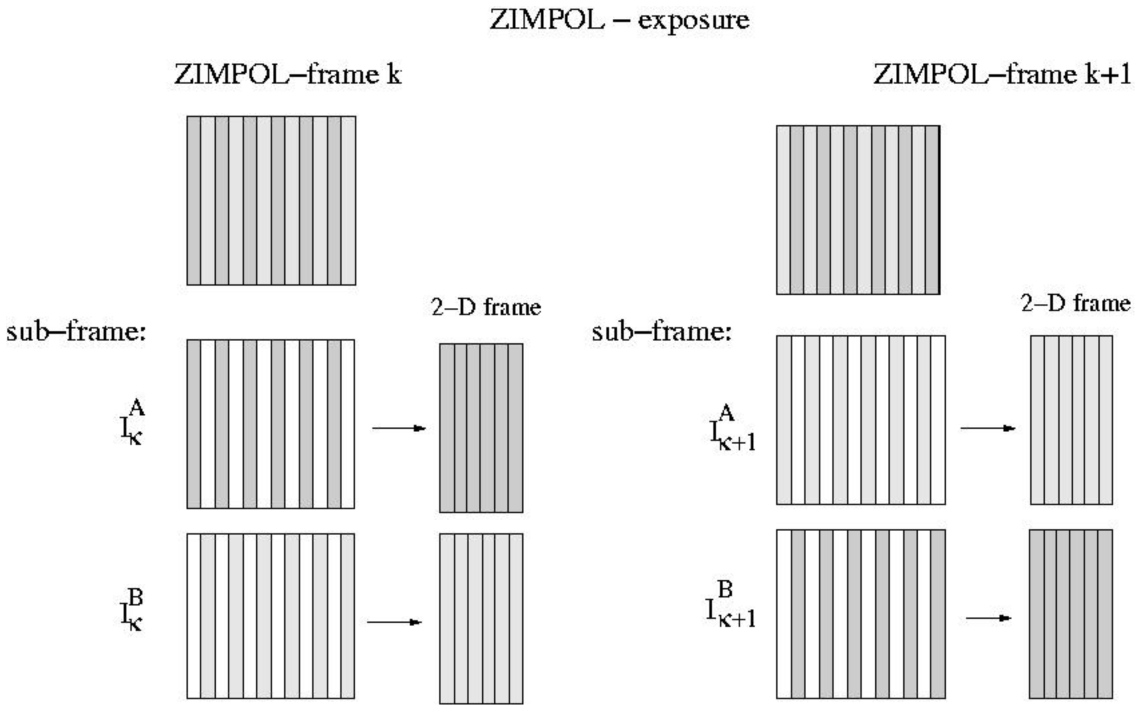


Figure 4.2: Schema of one single ZIMPOL *exposure* for the example of NDIT=1, leading to the output of two consecutive *frames* from each CCD. The left side is the kth ZIMPOL frame recorded at phase one of the double-phase mode, whereas the right side (frame $k+1$) has been recorded in the second phase. Each *frame* contains two interlaced *sub-frames*, storing two complementary polarization component images. Since the *frame* is square, extracting the two *sub-frames* yields two images with a 1:2 aspect ratio; a circle will be imaged as an ellipse). Dark grey means more intensity than light grey; white means that these columns contain no scientific data (but only noise).

binning leads to the software pixels of $30 \times 30\,\mu$m size which are read out. Later in the data reduction binning is applied anew, shown here with the largest filled box. A circle with the diameter of diffraction limited resolution is over-plotted to visualize the relations.

4.18.4 Dithering

Dithering is foreseen for all ZIMPOL observations. It will be implemented by keeping the telescope pointed at a fixed position on the sky and producing a series of movements of the tilt- and tip/tilt mirrors in front of the ZIMPOL cameras providing a series of x, y-shifts of the field of view by a certain number of pixels.

The proposed idea is to enter a number of dithering positions (NDITHER) and the individual x, y offsets $(\Delta x_i, \Delta y_i)$ in the P2PP. The input NDITHER and the number of $(\Delta x, \Delta y)$-pairs are compared, and an error is signalized if they do not match. The offsets $(\Delta x_i, \Delta y_i)$ can be given in arcseconds or in pixel numbers (units to be selected from a menu bar). The numbers are calculated by the INS and the commands are given to the motors of the tilt- and tip/tilt mirrors. It shall also be possible to select predefined dithering positions, e.g. 9 (or 25) positions with the pointing at the center and the remaining 8 (or 24) positions aligned in a grid around the pointing position with offsets of e.g. 2 pixels. For each NDITHER position both HWP2 positions with NDIT frames each can be taken at the fixed position of FOV (telescope and tilt- and tip/tilt mirrors). The observation sequence is according to the “Observations modes and sequences” (RD4) the following:

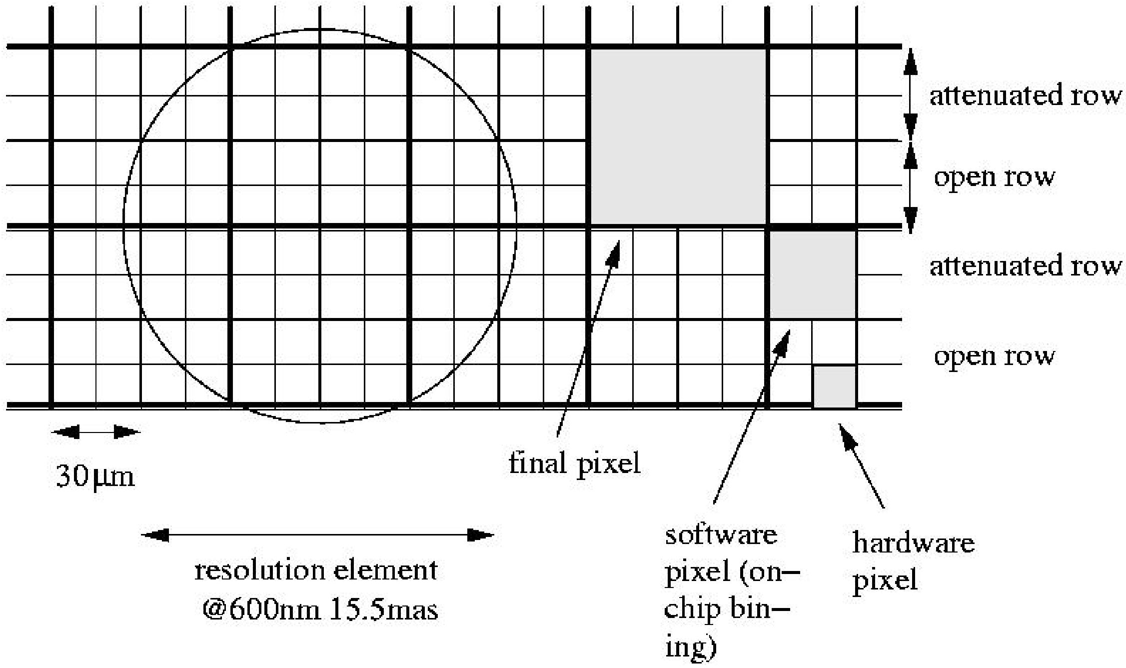


Figure 4.3: Detail of one ZIMPOL CCD with shown resolution element λ/D, the hardware pixel, the 2×2 binned software pixel and the final reduction pixel of 4×4 hardware pixels.

$$\text{NDIT frames} \Longrightarrow \text{HWP2 flip } (+45°) \Longrightarrow \text{NDITHER dithering positions.}$$

The following pseudo-code describes the procedure for a Q/I measurement (one exposure):

```
for iDither=0, NDITHER-1 do
    assume dithering position iDither
    set HWP2-offset angle to 0
    for iDIT=0, NDIT-1 do
        take 1 frame with the first mode of the double-phase mode
        take 1 frame with the second mode of the double-phase mode
    end for
    set HWP2-offset angle to 45
    for iDIT=0, NDIT-1 do
        take 1 frame with the first mode of the double-phase mode
        take 1 frame with the second mode of the double-phase mode
    end for
end for
```

4.18.5 Two-phase mode

In polarimetric detector modes, two more calibrations must be applied to the data after dark subtraction and intensity flat-fielding. As described in Section 4.18.2, each ZIMPOL frame k consists of two interlaced sub-frames $i^{\rm A}(k)$, $i^{\rm B}(k)$ that each store one polarization component, e.g. Q_+, Q_-. From one frame to the next, the assignment of polarization components to sub-frames is reversed by switching the phase of the demodulation cycle from 0 to π, whereas the detector's fixed pattern noise (FPN) remains unchanged. This property is exploited to remove the fixed pattern noise:

$$Q^{(0)} = i_k^{\mathrm{A}} - i_k^{\mathrm{B}} = \frac{1}{2}\left(\left(+Q + FPN^{\mathrm{A}}\right) - \left(-Q + FPN^{\mathrm{B}}\right)\right), \tag{4.11}$$

$$Q^{(\pi)} = i_{k+1}^{\mathrm{A}} - i_{k+1}^{\mathrm{B}} = \frac{1}{2}\left(\left(-Q + FPN^{\mathrm{A}}\right) - \left(+Q + FPN^{\mathrm{B}}\right)\right), \tag{4.12}$$

$$Q = \frac{1}{2}\left(Q^{(0)} - Q^{(\pi)}\right). \tag{4.13}$$

The fixed pattern noise of a given pixel has been assumed to be constant here. In reality, it can depend on the flux of the pixel to some degree. If the pixel flux is different between the two polarization components, i.e. if $Q \neq 0$, some residual fixed pattern noise will survive this process. Therefore, it is imperative to keep the overall background polarization of the science image as low as possible. The ZIMPOL instrument uses a rotatable and tiltable glass plate to compensate the background polarization in real-time for this purpose.

4.19 Specific ZIMPOL detector calibration

We distinguish between two quite separate kinds of calibration: The detector calibration, which attempts to remove all detector imperfections from the read-out photocharge images and reconstruct the actual intensity distribution incident on the detectors (and is therefore locked to the grid of the detector pixels); and the Stokes vector calibration, which attempts to reconstruct the scientific Stokes vector coming in from the sky on the basis of the intensity distributions that reach the detector after propagation through the instrument (and is therefore locked to the coordinate system of the sky image).

4.19.1 Polarization flatfield and modulation / demodulation efficiency

In order to remove any remaining spurious polarization from the detector, the so-called polarization flatfield (PFF) is recorded under flat, low-polarization illumination using two-phase mode. The resulting Q/I image must be subtracted from the Q/I science image.

Furthermore, the modulation / demodulation polarimetric efficiency of the ZIMPOL instrument (the $Q \rightarrow Q$ element of its Mueller matrix) is dominated by effects of imperfect demodulation; therefore it is corrected pixel-wise as part of the detector calibration rather than as part of the Stokes vector calibration. The modulation / demodulation efficiency (MDE) is recorded under 100% polarized flat illumination using two-phase mode. The Q/I science image is then divided by the resulting Q/I efficiency frame to remove those effects.

Therefore, the clean polarization image can be obtained as follows:

$$(Q/I)_{\mathrm{clean}} = \frac{(Q/I) - (Q/I)_{\mathrm{PFF}}}{(Q/I)_{\mathrm{MDE}}}. \tag{4.14}$$

This procedure is equivalent to the one performed in the intensity regime in Equation **??**, with the polarization flatfield taking the role of the dark current, and the modulation / demodulation efficiency taking the role of the intensity flatfield. (Obviously, the term "polarization flatfield" is a bit of a misnomer. "Polarized fixed-pattern noise" would have been more appropriate.)

4.20 Stokes vector calibration

The following pages describe how the original Stokes polarization image before the telescope can be inferred from the intensity images measured by the ZIMPOL cameras.

4.20.1 Signal switching

The first stage of Stokes vector calibration can be achieved by signal switching. During a Q or U observation, the half-wave plate HWP2 is repeatedly rotated by $\pm 45°$, causing the local (Q, U) polarization to change sign $(-Q, -U)$. On the other hand, all purely instrumental contributions to the Stokes vector downstream of HWP2 remain unchanged. Therefore, subtracting the two resulting co-added frames from each other results in a "clean" Stokes polarization image in which the instrumental contributions cancel out:

$$Q^{0°} = Q^{\mathrm{real}} + Q^{\mathrm{instr}} \qquad Q^{45°} = -Q^{\mathrm{real}} + Q^{\mathrm{instr}}$$

$$\Longrightarrow \quad Q^{\mathrm{real}} = \frac{1}{2}\left(Q^{0°} - Q^{45°}\right), \tag{4.15}$$

$$U^{22.5°} = U^{\mathrm{real}} + U^{\mathrm{instr}} \qquad U^{67.5°} = -U^{\mathrm{real}} + U^{\mathrm{instr}}$$

$$\Longrightarrow \quad U^{\mathrm{real}} = \frac{1}{2}\left(U^{22.5} - U^{67.5°}\right). \tag{4.16}$$

Note that any spurious *manipulation* of the (Q, U) polarization downstream of HWP2, i.e. all instrumental effects proportional to the incoming real polarization, survive this calibration step unscathed. They must be removed with the help of previously measured calibration matrices, as detailed in the following sections.

4.20.2 Mueller matrix instrument model

The original plan for the polarimetric calibration of SPHERE ZIMPOL observations was to model the telescope, the adaptive optics and coronagraph subsection (AO&C), and the ZIMPOL instrument as three Mueller matrices $\mathbf{T}, \mathbf{C}, \mathbf{Z}$, through which the true Stokes vector $\vec{S}_0$ is propagated to form the measured Stokes vector $\vec{S}_m$:

$$\vec{S}_m = \mathbf{Z} \cdot \mathbf{C} \cdot \mathbf{T} \cdot \vec{S}_0 =: \mathbf{M} \cdot \vec{S}_0. \tag{4.17}$$

If the significant elements of the three matrices can be obtained from calibration measurements, the true Stokes vector can be reconstructed from the measured one by means of the calibration matrix $\mathbf{M^{-1}}$:

$$\mathbf{M}^{-1} = (\mathbf{ZCT})^{-1}, \qquad \vec{S}_0 = \mathbf{M^{-1}} \cdot \vec{S}_m. \tag{4.18}$$

However, this ansatz is not compatible with the reality of the SPHERE ZIMPOL instrument. The polarimeter is only capable of measuring one Stokes polarization (Q or U) at a given time. When switching among Q and U measurements, a half-wave plate (HWP2) is rotated between the telescope and the AO&C subsection, effectively *changing the Mueller matrix chain* and therefore rendering the inversion problem less trivial. Namely, during U measurements, the polarization corresponding to Stokes U on the science target is flipped into the Q channel, and vice versa. The Q and U components of the science image are therefore both propagated through the Q channel of the **C** and **Z** matrices when they are being measured, and through the U channel during the other measurements.

In order to account for the nature of polarization measurement in the instrument, we add a fourth Mueller matrix to the model to represent the variable effect of the half-wave plate HWP2.

For Q measurements, the model is

$$\vec{S}_m = \begin{pmatrix} I \\ Q_m \\ \circ \\ \circ \end{pmatrix} = \mathbf{Z} \cdot \mathbf{C} \cdot \mathbf{H} \cdot \mathbf{T} \cdot \vec{S}_0 = \mathbf{Z} \cdot \mathbf{C} \cdot \mathbf{H} \cdot \mathbf{T} \cdot \begin{pmatrix} I \\ Q_0 \\ U_0 \\ \circ \end{pmatrix}, \tag{4.19}$$

and for U measurements:

$$\vec{S}_m = \begin{pmatrix} I \\ U_m \\ \circ \\ \circ \end{pmatrix} = \mathbf{Z} \cdot \mathbf{C} \cdot \tilde{\mathbf{H}} \cdot \mathbf{T} \cdot \vec{S}_0 = \mathbf{Z} \cdot \mathbf{C} \cdot \tilde{\mathbf{H}} \cdot \mathbf{T} \cdot \begin{pmatrix} I \\ Q_0 \\ U_0 \\ \circ \end{pmatrix}. \tag{4.20}$$

All Stokes vectors are assumed to be normalized by their intensity I. Since the science target is not expected to have a circularly polarized component, we set V_m to zero. The open bullet ($\circ$) is used to represent zero here in order to emphasize the non-zero elements.

Although the equation is no longer as elegant as before, the three measured values I, Q_m, U_m are still independent linear combinations of the three science values I, Q_0, U_0, which can be described by a Mueller matrix **M**. Therefore, there still exists a Mueller matrix $\mathbf{M}^{-1}$ which can reduce the measured Stokes vector into clean science data.

4.20.3 Telescope Matrix

The propagation through the telescope optics is represented by the following Mueller matrix:

$$\mathbf{T} = \begin{pmatrix} 1 & \circ & \circ & \circ \\ t_{IQ} & t_{QQ} & t_{UQ} & \circ \\ t_{IU} & t_{QU} & t_{UU} & \circ \\ \circ & \circ & \circ & \circ \end{pmatrix}. \tag{4.21}$$

The fourth column is set to zero, since there is no circular component in the input Stokes vector $\vec{S}_0$. Furthermore, we set the fourth *row* to zero as well; although $Q, U \to V$ crosstalks are possible, the resulting V polarization will not be inverted by HWP2 switching (unlike the Q and U signals), thus its contribution to the measured signal will have canceled out by the time the Stokes calibration process is applied.

Apart from the elevation angle h, the matrix **T** is also a function of the parallactic angle p at the time of observation, since the coordinate system of the sky target is offset from the telescope's coordinate system by this angle. If we assume that the telescope matrix $\mathbf{T}_h$ as measured in its own coordinate system (defined by the M3 tilt axis) is independent of p for a given h, we can model **T** for an arbitrary p as:

$$\mathbf{T}(p, h) = \mathbf{R}(-p)\mathbf{T}_h\mathbf{R}(p), \tag{4.22}$$

where $\mathbf{R}(p)$ represents a rotation in the linear polarization plane by the angle p:

$$\mathbf{R}(p) := \begin{pmatrix} 1 & \circ & \circ & \circ \\ \circ & \cos(2p) & \sin(2p) & \circ \\ \circ & -\sin(2p) & \cos(2p) & \circ \\ \circ & \circ & \circ & 1 \end{pmatrix}. \tag{4.23}$$

Thus, the knowledge of $\mathbf{T}_h$ for a representative set of elevation angles h is enough to derive the telescope matrix $\mathbf{T}$ for a given situation (p, h).

Note that the merit of this particular dependence might remain purely academic, though, since the use of HWP1 between M3 and M4 should ensure a $\mathbf{T}$ almost identical to the unit matrix at any time.

Obtaining the matrix elements: The elements IQ, IU of $\mathbf{T}_h$ are provided by the recipe sph_-zpl_telescope_offset, the elements QQ, UU by the recipe sph_zpl_telescope_zeropoint_angle. The elements QU, UQ of $\mathbf{T}_h$ are not currently measured by a recipe and are therefore set to zero.

4.20.4 HWP2 Matrices

During a Stokes Q measurement, HWP2 is oriented parallel to a Q direction and therefore does not alter the incoming Q polarization, but the U polarization is mirrored. During the Stokes U measurement, HWP2 is oriented between the Q and U directions and therefore simply exchanges the incoming Q and U polarizations. The matrices $\mathbf{H}$ (for Q measurements) and $\tilde{\mathbf{H}}$ (for U measurements) are therefore

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \tilde{\mathbf{H}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.24}$$

During half of the observation time, the HWP2 position angle will be offset by 45° with respect to its default position, inverting the linear polarizations at this stage $((Q, U) \rightarrow (-Q, -U))$. This need not be included in these matrices, since the inverted frames will already have been subtracted from the regular frames by the time the Stokes calibration is applied.

4.20.5 Adaptive Optics & Coronagraph Matrix

The half-wave plates HWP2 and HWPZ ensure that the measured polarization component (in the Q channel) always passes through the AO&C subsection such that it aligns with the orientation of the tilted derotator optics. This means that the other component (in the U channel) is particularly prone to crosstalk into circular polarization (c_{UV}).

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{QQ} & c_{UQ} & 0 \\ 0 & c_{QU} & c_{UU} & 0 \\ 0 & c_{QV} & c_{UV} & 0 \end{pmatrix}. \tag{4.25}$$

Note that while the elements c_{V*} may not be zero in reality, they are irrelevant to this calibration process, since the fourth row of $\mathbf{HT}$ and $\tilde{\mathbf{H}}\mathbf{T}$ are zero. This will eliminate those elements from the products $\mathbf{CHT}$ and $\mathbf{C}\tilde{\mathbf{H}}\mathbf{T}$. Furthermore, the elements c_{IQ} and c_{IU} must be set to zero, since HWP2 switching eliminates all static instrumental polarization from the measured data.

Again, this is because all contributions to measured polarization from V polarization upstream of HWP2 will have canceled out during signal switching. However, the V polarization generated by the $Q, U \rightarrow V$ crosstalks in this matrix are based on the Q, U polarizations after HWP2, and are therefore resistant to signal switching.

The matrix $\mathbf{C}$ is non-trivially dependent on the position angle a of the derotator optics.

Obtaining the matrix elements: The elements c_{QQ}, c_{UU} are provided by the recipe sph_zpl_-aoc_efficiency, the elements $c_{QU}, c_{QV}, c_{UQ}, c_{UV}$ by the recipe sph_zpl_aoc_crosstalk.

4.20.6 ZIMPOL Matrix

The ZIMPOL instrument is only capable of measuring the Stokes parameters (I, Q), therefore the other two rows of the matrix are empty. (Again, the measured polarization component always travels through the Q channel, regardless of whether it represents the Q or U component of the science image.) Nevertheless, incoming U and V polarization can end up in the Q channel due to crosstalks. The instrument polarization of ZIMPOL (z_{IQ}) can be neglected here, since it is effectively eliminated by the two-phase detector mode and the signal-switching calibration.

$$\mathbf{Z} = \begin{pmatrix} 1 & \circ & \circ & \circ \\ \circ & z_{QQ} & z_{UQ} & z_{VQ} \\ \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ \end{pmatrix}. \tag{4.26}$$

Obtaining the matrix elements: Note that the science frames will already have been corrected for ZIMPOL modulation / demodulation efficiency factor (the absolute $Q \rightarrow Q$ efficiency of ZIMPOL) on a pixel-by-pixel basis before this Mueller matrix calibration is applied. All $* \rightarrow Q$ elements of $\mathbf{Z}$ must therefore be divided by the mean modulation / demodulation efficiency ($= z_{QQ}$) to avoid correcting the science data twice for the same factor.

Therefore, z_{QQ} must be set to 1 in this matrix, The elements z_{UQ} and z_{VQ} are directly provided by the recipe sph_zpl_zimpol_crosstalk; the measured frames used in this recipe are already corrected for modulation / demodulation efficiency before the calibration products are extracted.

4.20.7 Inversion of an instantaneous measurement

4.20.7.1 Q measurement

We begin by multiplying the first two matrices:

$$\mathbf{H} \cdot \mathbf{T} = \begin{pmatrix} 1 & \circ & \circ & \circ \\ t_{IQ} & t_{QQ} & t_{UQ} & \circ \\ -t_{IU} & -t_{QU} & -t_{UU} & \circ \\ \circ & \circ & \circ & \circ \end{pmatrix}. \tag{4.27}$$

Including the third matrix, we get:

$$\mathbf{C} \cdot \mathbf{H} \cdot \mathbf{T} =$$
$$\begin{pmatrix} 1 & \circ & \circ & \circ \\ t_{IQ}c_{QQ} - t_{IU}c_{UQ} & t_{QQ}c_{QQ} - t_{QU}c_{UQ} & t_{UQ}c_{QQ} - t_{UU}c_{UQ} & \circ \\ t_{IQ}c_{QU} - t_{IU}c_{UU} & t_{QQ}c_{QU} - t_{QU}c_{UU} & t_{UQ}c_{QU} - t_{UU}c_{UU} & \circ \\ t_{IQ}c_{QV} - t_{IU}c_{UV} & t_{QQ}c_{QV} - t_{QU}c_{UV} & t_{UQ}c_{QV} - t_{UU}c_{UV} & \circ \end{pmatrix} \tag{4.28}$$

Due to the empty rows and columns of the matrices $\mathbf{Z}$ and $\mathbf{C} \cdot \mathbf{H} \cdot \mathbf{T}$, the total measurement matrix $\mathbf{X}$ has the following shape:

$$\mathbf{X} := \mathbf{Z} \cdot \mathbf{C} \cdot \mathbf{H} \cdot \mathbf{T} = \begin{pmatrix} 1 & \circ & \circ & \circ \\ x_{IQ} & x_{QQ} & x_{UQ} & \circ \\ \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ \end{pmatrix}, \tag{4.29}$$

where

$$\begin{aligned} x_{IQ} &= z_{QQ}(t_{IQ}c_{QQ} - t_{IU}c_{UQ}) \\ &+ z_{UQ}(t_{IQ}c_{QU} - t_{IU}c_{UU}) \\ &+ z_{VQ}(t_{IQ}c_{QV} - t_{IU}c_{UV}), \\ \\ x_{QQ} &= z_{QQ}(t_{QQ}c_{QQ} - t_{QU}c_{UQ}) \\ &+ z_{UQ}(t_{QQ}c_{QU} - t_{QU}c_{UU} \\ &+ z_{VQ}(t_{QQ}c_{QV} - t_{QU}c_{UV}), \\ \\ x_{UQ} &= z_{QQ}(t_{UQ}c_{QQ} - t_{UU}c_{UQ}) \\ &+ z_{UQ}(t_{UQ}c_{QU} - t_{UU}c_{UU}) \\ &+ z_{VQ}(t_{UQ}c_{QV} - t_{UU}c_{UV}). \end{aligned} \tag{4.30}$$

Using these elements, the Q measurement can be summed up as:

$$\mathbf{X}\begin{pmatrix} I \\ Q_0 \\ U_0 \\ \circ \end{pmatrix} = \begin{pmatrix} I \\ Q_m \\ \circ \\ \circ \end{pmatrix} \implies Q_m = x_{IQ}I + x_{QQ}Q_0 + x_{UQ}U_0. \tag{4.31}$$

4.20.7.2 *U* measurement

We begin by multiplying the first two matrices:

$$\tilde{\mathbf{H}} \cdot \mathbf{T} = \begin{pmatrix} 1 & \circ & \circ & \circ \\ t_{IU} & t_{QU} & t_{UU} & \circ \\ t_{IQ} & t_{QQ} & t_{UQ} & \circ \\ \circ & \circ & \circ & \circ \end{pmatrix}. \tag{4.32}$$

Including the third matrix, we get:

$$\mathbf{C} \cdot \tilde{\mathbf{H}} \cdot \mathbf{T} = \begin{pmatrix} 1 & \circ & \circ & \circ \\ t_{IU}c_{QQ} + t_{IQ}c_{UQ} & t_{QU}c_{QQ} + t_{QQ}c_{UQ} & t_{UU}c_{QQ} + t_{UQ}c_{UQ} & \circ \\ t_{IU}c_{QU} + t_{IQ}c_{UU} & t_{QU}c_{QU} + t_{QQ}c_{UU} & t_{UU}c_{QU} + t_{UQ}c_{UU} & \circ \\ t_{IU}c_{QV} + t_{IQ}c_{UV} & t_{QU}c_{QV} + t_{QQ}c_{UV} & t_{UU}c_{QV} + t_{UQ}c_{UV} & \circ \end{pmatrix} \tag{4.33}$$

Due to the empty rows and columns of the matrices $\mathbf{Z}$ and $\mathbf{C} \cdot \tilde{\mathbf{H}} \cdot \mathbf{T}$, the total measurement matrix $\tilde{\mathbf{X}}$ has the following shape:

$$\tilde{\mathbf{X}} := \mathbf{Z} \cdot \mathbf{C} \cdot \tilde{\mathbf{H}} \cdot \mathbf{T} = \begin{pmatrix} 1 & \circ & \circ & \circ \\ \tilde{x}_{IQ} & \tilde{x}_{QQ} & \tilde{x}_{UQ} & \circ \\ \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ \end{pmatrix}, \tag{4.34}$$

where

$$\begin{aligned}
\tilde{x}_{IQ} &= z_{QQ}(t_{IU}c_{QQ} + t_{IQ}c_{UQ}) \\
&+ z_{UQ}(t_{IU}c_{QU} + t_{IQ}c_{UU}) \\
&+ z_{VQ}(t_{IU}c_{QV} + t_{IQ}c_{UV}), \\
\\
\tilde{x}_{QQ} &= z_{QQ}(t_{QU}c_{QQ} + t_{QQ}c_{UQ}) \\
&+ z_{UQ}(t_{QU}c_{QU} + t_{QQ}c_{UU}) \\
&+ z_{VQ}(t_{QU}c_{QV} + t_{QQ}c_{UV}), \\
\\
\tilde{x}_{UQ} &= z_{QQ}(t_{UU}c_{QQ} + t_{UQ}c_{UQ}) \\
&+ z_{UQ}(t_{UU}c_{QU} + t_{UQ}c_{UU}) \\
&+ z_{VQ}(t_{UU}c_{QV} + t_{UQ}c_{UV}).
\end{aligned} \tag{4.35}$$

Using these elements, the U measurement can be summed up as:

$$\tilde{\mathbf{X}} \begin{pmatrix} I \\ Q_0 \\ U_0 \\ \circ \end{pmatrix} = \begin{pmatrix} I \\ U_m \\ \circ \\ \circ \end{pmatrix} \implies U_m = \tilde{x}_{IQ} I + \tilde{x}_{QQ} Q_0 + \tilde{x}_{UQ} U_0. \tag{4.36}$$

4.20.7.3 Inversion

Combining Equations 4.31 and 4.36 and normalizing each side by the intensity, we end up with two linear equations for the two unknown degrees of polarization $(Q/I)_0, (U/I)_0$, written in the two known degrees of polarization $(Q/I)_m, (U/I)_m$. We can write this relationship with a 3×3 matrix:

$$\begin{pmatrix} 1 \\ (Q/I)_m \\ (U/I)_m \end{pmatrix} = \begin{pmatrix} 1 & \circ & \circ \\ x_{IQ} & x_{QQ} & x_{UQ} \\ \tilde{x}_{IQ} & \tilde{x}_{QQ} & \tilde{x}_{UQ} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ (Q/I)_0 \\ (U/I)_0 \end{pmatrix} =: \mathbf{M} \cdot \begin{pmatrix} 1 \\ (Q/I)_0 \\ (U/I)_0 \end{pmatrix}. \tag{4.37}$$

The final calibration matrix $\mathbf{M}^{-1}$ is therefore the inverse of the matrix above:

$$\begin{pmatrix} 1 \\ (Q/I)_0 \\ (U/I)_0 \end{pmatrix} = \mathbf{M}^{-1} \cdot \begin{pmatrix} 1 \\ (Q/I)_m \\ (U/I)_m \end{pmatrix}. \tag{4.38}$$

Neither the input nor the output Stokes vector contain any circular polarization, but second-order crosstalks in the fashion of $Q \to V \to U$ have been taken into account.

4.20.7.4 Solution for long observations

The inversion presented in the previous section assumes that the parameter-dependen matrices T and C remain constant throughout the measurement. In reality, however, the elevation angle h, the parallactic angle p and the derotator position angle a evolve on the timescale of Q and U measurements.

This problem can be solved as follows. Let $\langle\cdot\rangle$ denote the unweighted[2] average over the whole set

[2]During frame recombination, each frame is divided by its integration time and therefore has equal weight in the final co-added image.

of N frames taken during the observation time:

$$\langle \mathbf{X} \rangle = \frac{1}{N} \sum_{i=1}^{N} \mathbf{X}(h(t_i), p(t_i), a(t_i)), \qquad \langle Q_m \rangle = \frac{1}{N} \sum_{i=1}^{N} Q_m(t_i), \tag{4.39}$$

and likewise for $\langle \tilde{\mathbf{X}} \rangle, \langle U_m \rangle$. Then, thanks to the *linearity of Mueller matrix calculus*, Equations 4.31 and 4.36 can be rewritten with the time averaged values:

$$\langle Q_m \rangle = \langle x_{IQ} \rangle I + \langle x_{QQ} \rangle Q_0 + \langle x_{UQ} \rangle U_0, \qquad \langle U_m \rangle = \langle \tilde{x}_{IQ} \rangle I + \langle \tilde{x}_{QQ} \rangle Q_0 + \langle \tilde{x}_{UQ} \rangle U_0. \tag{4.40}$$

Therefore, Equation 4.37 becomes

$$\begin{pmatrix} 1 \\ \langle (Q/I)_m \rangle \\ \langle (U/I)_m \rangle \end{pmatrix} = \begin{pmatrix} 1 & \circ & \circ \\ \langle x_{IQ} \rangle & \langle x_{QQ} \rangle & \langle x_{UQ} \rangle \\ \langle \tilde{x}_{IQ} \rangle & \langle \tilde{x}_{QQ} \rangle & \langle \tilde{x}_{UQ} \rangle \end{pmatrix} \cdot \begin{pmatrix} 1 \\ (Q/I)_0 \\ (U/I)_0 \end{pmatrix} =: \mathbf{M}_{\text{global}} \cdot \begin{pmatrix} 1 \\ (Q/I)_0 \\ (U/I)_0 \end{pmatrix}. \tag{4.41}$$

Although the averaging $\langle \cdot \rangle$ has a different concrete meaning for the second and third row in this equation (since the Q and U values are derived from mutually exclusive sets of frames), the definition is consistent within each row. This allows us to reduce the global time-averaged Q/I and U/I masterframes in a single step:

$$\begin{pmatrix} 1 \\ (Q/I)_0 \\ (U/I)_0 \end{pmatrix} = \mathbf{M}_{\text{global}}^{-1} \cdot \begin{pmatrix} 1 \\ \langle (Q/I)_m \rangle \\ \langle (U/I)_m \rangle \end{pmatrix}. \tag{4.42}$$

Chapter 5

Instrument Data Description

In this section we describe the raw data, including the for DRH relevant DPR keywords for each data type. All valid combinations of FITS DPR keywords are identified, listed and identified with corresponding recipes which need to use them as input.

For each of these data structures the basic data type is a FITS file with an image in the HDU corresponding to the full 2048x2048 pixel region of the detector and no extensions. The keywords in the header of the FITS file depend strongly on the data structure represented. The following table lists the keywords for each of the data structures for technical, science and monitoring calibrations.

5.1 General Data Layout

A raw SPHERE file always has the images stored in the primary FITS data unit. Please see the FDR document for a table that lists the raw data types, the corresponding calibrations (names as in the calibration plan) etc. In sections 9-11 of this manual the data types, keywords etc are listed for each recipe.

5.2 Imaging Frames

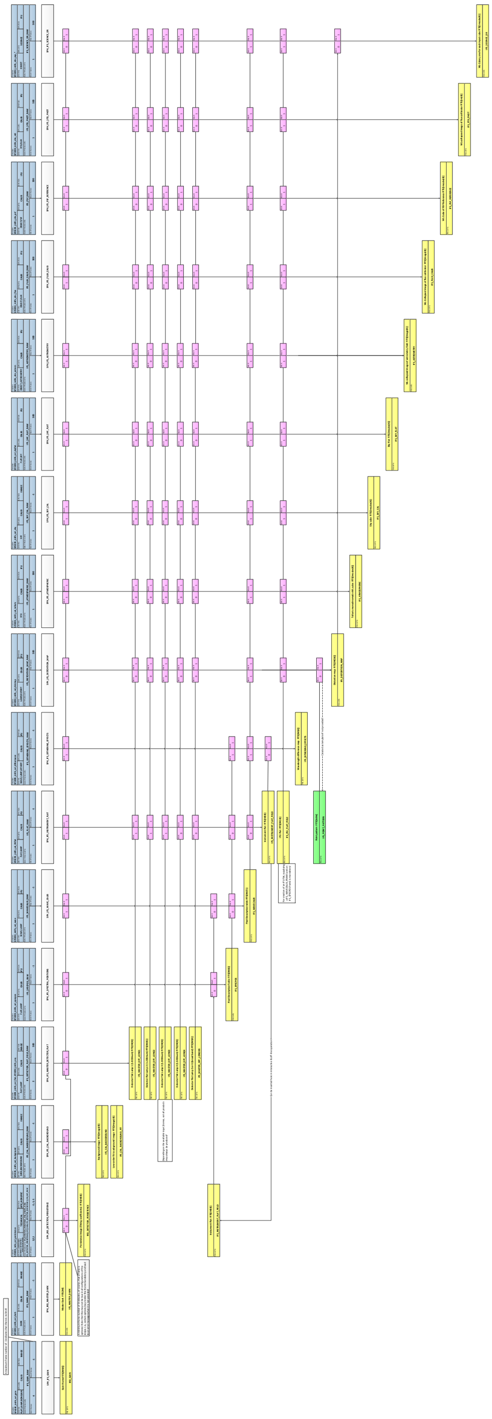
The raw imaging frames for IFS and IRDIS all contain in total 2048x2048 pixels. For IRDIS all raw frames that are obtained with calibrations that illuminate the detector through the IRDIS optical path, only an area of 2048x1024 pixels is used. This again is split in two parts for classical imaging and DBI modes.

5.2.1 Image Coordinate System

In several places the SPHERE recipes report coordinates, in particular of a reported frame center, determined star position, or similar. These positions are communicated to the user by the means of header keywords or dedicated product files, either as FITS tables or, optionally, simply ascii output. When comparing these coordinates to values derived by the means of using other tools, care should be taken about the coordinate system in use. Many ways exist to describe locations in image frames, and while the SPHERE pipeline uses pixel coordinates, there are also different ways of defining the pixel grid coordinate system. The FITS standard has its own definition, and the various FITS viewing tools and scripting (and other programming) languages follow different standards.

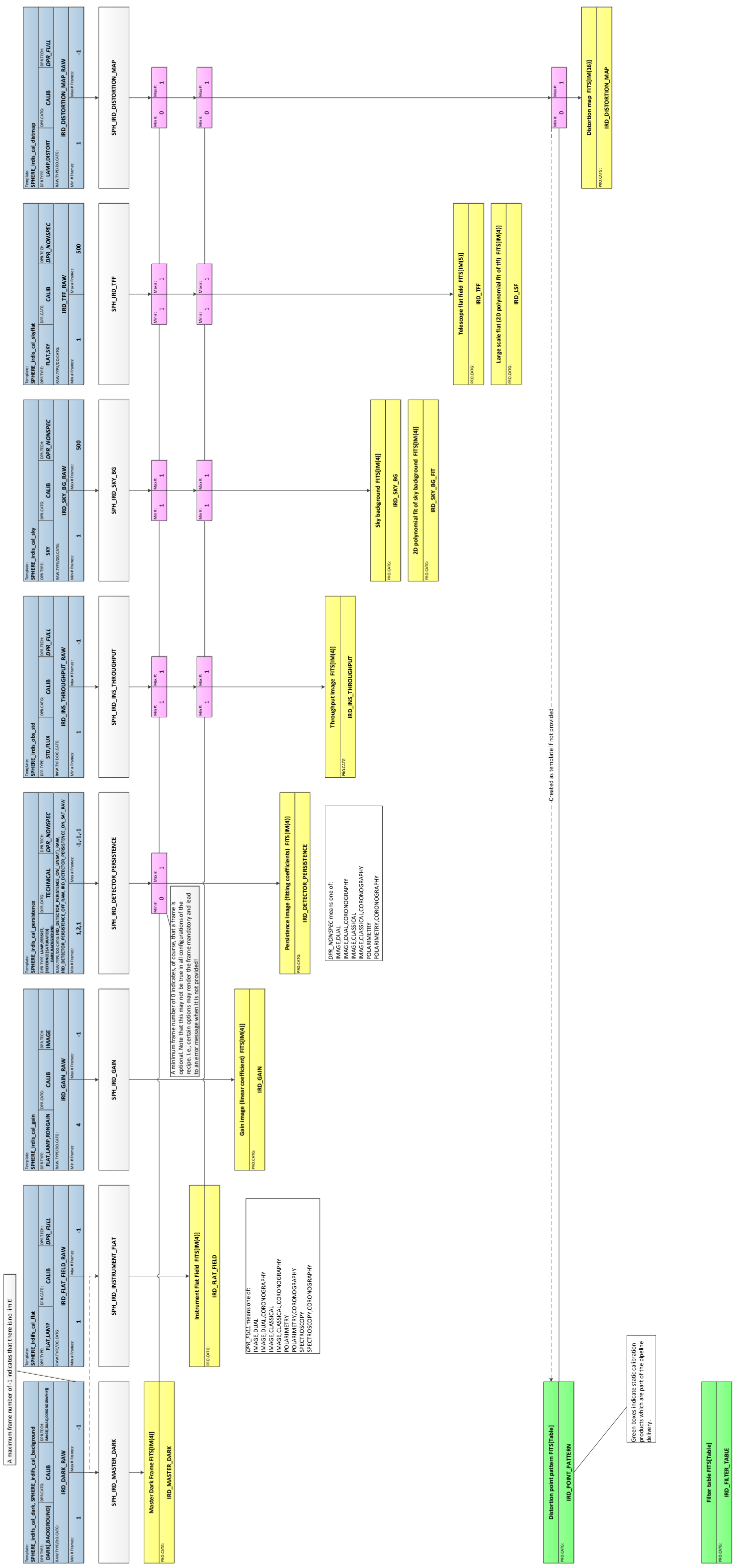
Coordinates reported (or used as input) by the SPHERE pipeline always refer to the following scheme: For an image consisting of NxM pixels, the coordinate (0,0) refers to the lower left corner of the lower left pixel. The coordinate (N,M) refers to the upper right corner of the upper right pixel. The midpoint of the lower left pixel has the coordinate (0.5,0.5), and the image midpoint (N/2,M/2).

5.3 IFS Calibration Data

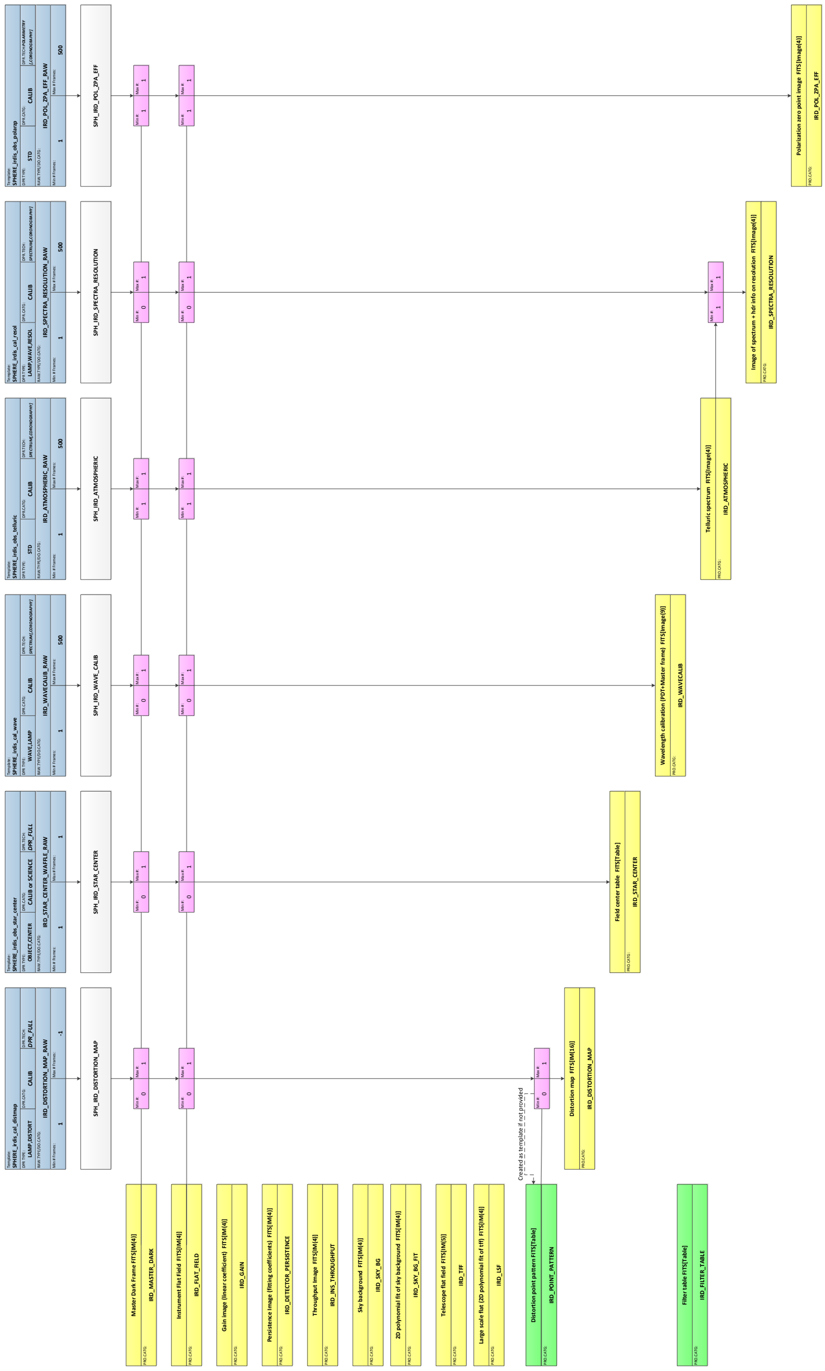


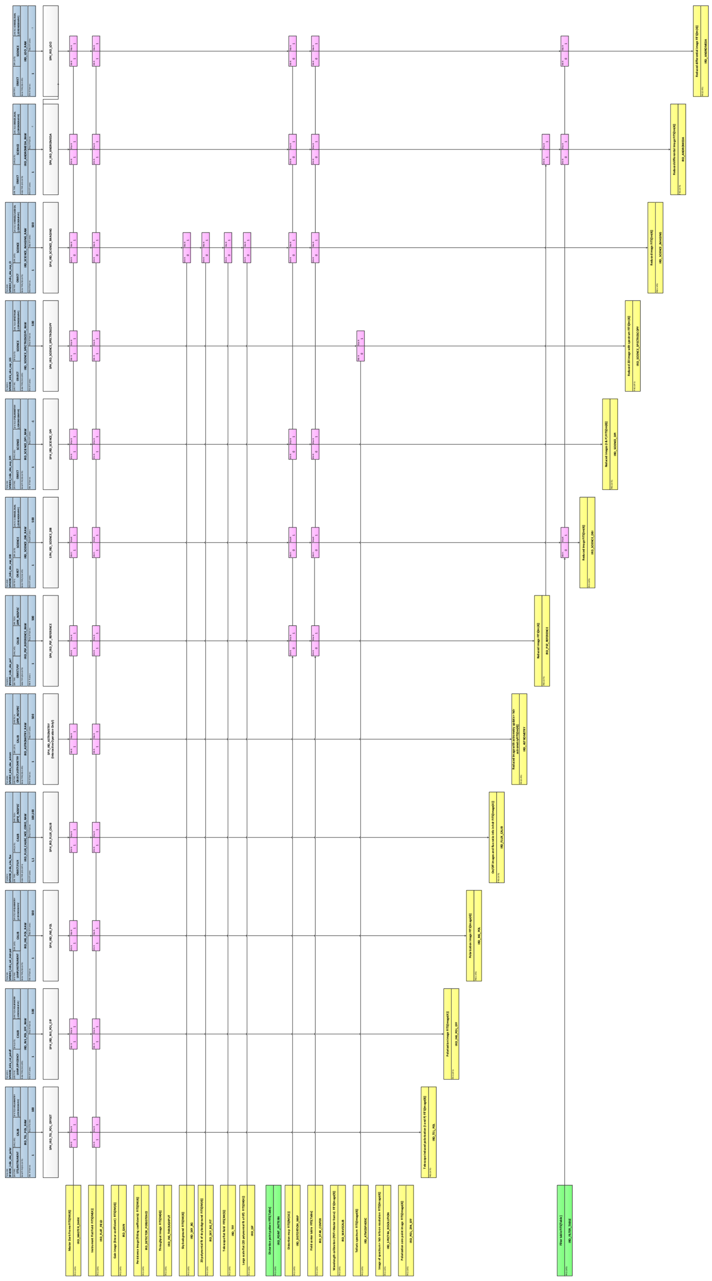
The graph shows the base configuration of the IFS pipeline. The list of recipe products is not necessarily complete under all circumstances. Also, additional optional may exist which are not shown here. Obviously, the graph is best viewed at high magnification in the electronic version of the document!

5.4 IRDIS Data



The graph shows the initital part of the base configuration of the IRDIS pipeline. The list of recipe products is not necessarily complete under all circumstances. Also, additional optional may exist which are not shown here. Obviously, the graph is best viewed at high magnification in the electronic version of the document!





5.5 ZIMPOL Data

	DO CATG	CAL ID	Template	Recipe	DPR.CATG	DPR.TYPE	DPR.TECH	FITS Cards used	PROD.CATG
ZIMPOL									
Bias	ZPL_BIAS_RAW	ZIMPOL-TEC-01	SPHERE_zimpol_cal_bias_pol	sph_zpl_master_bias	CALIB	BIAS	POLARIMETRY		ZPL_MASTER_BIAS
Bias img	ZPL_BIAS_IMAGING_RAW	ZIMPOL-TEC-01	SPHERE_zimpol_cal_bias_img	sph_zpl_master_bias_imaging	CALIB	BIAS	IMAGE		ZPL_MASTER_BIAS_IMAGING
Dark	ZPL_DARK_RAW	ZIMPOL-TEC-02	SPHERE_zimpol_cal_dark_pol	sph_zpl_master_dark	CALIB	DARK	POLARIMETRY		ZPL_MASTER_DARK
Dark img	ZPL_DARK_IMAGING_RAW	ZIMPOL-TEC-02	SPHERE_zimpol_cal_dark_img	sph_zpl_master_dark_imaging	CALIB	DARK	IMAGE		ZPL_MASTER_DARK_IMAGING
Polar. Flat	ZPL_POL_FLAT_FIELD_RAW	ZIMPOL-TEC-03	SPHERE_zimpol_cal_polflat	sph_zpl_polarization_flat	CALIB	FLAT,TEST	POLARIMETRY		ZPL_POL_FLAT_FIELD
Intens. Flat	ZPL_INT_FLAT_FIELD_RAW **ZPL_IFF_RAW**	ZIMPOL-TEC-04	SPHERE_zimpol_cal_flat_pol	sph_zpl_intensity_flat	CALIB	FLAT,LAMP	POLARIMETRY		ZPL_INT_FLAT_FIELD **ZPL_IFF**
Intens. Flat img.	ZPL_INT_FLAT_FIELD_IMAGING_RAW **ZPL_IFF_IMG_RAW**	ZIMPOL-TEC-04	SPHERE_zimpol_cal_flat_img	sph_zpl_intensity_flat_imaging	CALIB	FLAT,LAMP	IMAGE		ZPL_INT_FLAT_FIELD_IMAGING **ZPL_IFF_IMG**
Sky Flat	ZPL_SCIENCE_P1_RAW	ZIMPOL-TEC-05	SPHERE_zimpol_cal_skyflat_pol	sph_zpl_science_p1	CALIB	FLAT,SKY	POLARIMETRY		ZPL_SCIENCE_P1_REDUCED
Sky Flat Img.	ZPL_SCIENCE_IMAGING_RAW	ZIMPOL-TEC-05	SPHERE_zimpol_cal_skyflat_img	sph_zpl_science_imaging	CALIB	FLAT,SKY	IMAGE		ZPL_SCIENCE_IMAGING_REDUCED
MoDem Eff.	ZPL_MODEM_EFF_RAW	ZIMPOL-TEC-06	SPHERE_zimpol_cal_modem_poleff	sph_zpl_modem_efficiency	CALIB	FLAT,POL100	POLARIMETRY		ZPL_MODEM_EFF
AO&C polarization efficiency	ZPL_AOC_EFF_RAW	ZIMPOL-MON-01	SPHERE_zimpol_tec_poleff	sph_zpl_aoc_efficiency	TECHNICAL	LAMP,POLEFF	POLARIMETRY		ZPL_AOC_EFF
AO&c polarization offset	ZPL_AOC_OFFSET_RAW	ZIMPOL-MON-02	SPHERE_zimpol_tec_instrpol	sph_zpl_aoc_offset	TECHNICAL	LAMP,POFFSET	POLARIMETRY		ZPL_AOC_OFFSET
AO&C polarization x-talk	ZPL_AOC_CROSSTALK_RAW	ZIMPOL-MON-03	SPHERE_zimpol_tec_polxtalk	sph_zpl_aoc_crosstalk	TECHNICAL	LAMP,XTALK	POLARIMETRY		ZPL_AOC_CROSSTALK
ZIMPOL Crosstalk	ZPL_ZIMPOL_CROSSTALK_RAW	ZIMPOL-MON-04	SPHERE_zimpol_tec_intxtalk	sph_zpl_zimpol_crosstalk	TECHNICAL	LAMP,XTALK,ZIMPOL	POLARIMETRY		ZPL_ZIMPOL_CROSSTALK
Telescope crosstalk	ZPL_SCIENCE_P1_RAW	SIMPOL-MON-05	SPHERE_zimpol_tec_telxtalk	**sph_zimpol_science_p1**	TECHNICAL	STD,POLHIGH,XTALK	POLARIMETRY		ZPL_SCIENCE_p1_REDUCED
Astrometric distortion	ZPL_DISTORTION_MAP_IMAGING_RAW **ZPL_DISTMAP_IMG_RAW**	ZIMPOL-MON-06	SPHERE_zimpol_cal_distmap	**N/A**	CALIB	LAMP,ASTROMETRY	POLARIMETRY		ZPL_DISTPRTION_MAP_IMAGING **ZPL_DISTMAP_IMG**
Astrometry	ZPL_SCIENCE_IMAGING_RAW	SIMPOL-SCI-01	SPHERE_zimpol_cal_astrom	sph_zpl_science_imaging	CALIB	STD;ASTROMETRY	IMAGE		ZPL_SCIENCE_IMAGING_REDUCED
Photometry (pol()	ZPL_SCIENCE_P1_RAW	ZIMPOL-SCI-02	SPHERE_zimpol_cal_flux_pol	sph_zpl_science_p1	CALIB	STD,FLUX	POLARIMETRY		ZPL_SCIENCE_P1_REDUCED
Photometry (img)	ZPL_SCIENCE_IMAGING_RAW	ZIMPOL-SCI-02	SPHERE_zimpol_cal_flux_img	sph_zpl_science_imaging	CALIB	STD,FLUX	IMAGE		ZPL_SCIENCE_IMAGING_REDUCED
Telescope polarization offset	ZPL_SCIENCE_P1_RAW	ZIMPOL-SCI-03	SPHERE_zimpol_cal_polar	**sph_zimpol_science_p1**	CALIB	STD,POL0	POLARIMETRY		ZPL_SCIENCE_P1_REDUCED
Telescope zero angle	ZPL_SCIENCE_P1_RAW	ZIMPOL-SCI-04	SPHERE_zimpol_cal_polarzp	**sph_zimpol_science_p1**	CALIB	STD,POLHIGH,XTALK	POLARIMETRY		ZPL_SCIENCE_P1_REDUCED
Imaging Exposure	ZPL_SCIENCE_IMAGING_RAW		SPHERE_zimpol_obs_exp_img	sph_zpl_science_imaging	SCIENCE	OBJECT	IMAGE		ZPL_SCIENCE_IMAGING_REDUCED
P1 Exposure	ZPL_SCIENCE_P1_RAW		SPHERE_zimpol_obs_exp_polp1	sph_zpl_science_p1	SCIENCE	OBJECT	POLARIMETRY,P1		ZPL_SCIENCE_P1_REDUCED
P2P3 Exposure	ZPL_SCIENCE_P2P3_RAW		SPHERE_zimpol_obs_exp_polp2p3	sph_zpl_science_p2p3	SCIENCE	OBJECT	POLARIMETRY,P2P3		ZPL_SCIENCE_P12P3_REDUCED
Ghost calibration	ZPL_SCIENCE_IMAGING_RAW	ZIMPOL-MON-07	SPHERE_zimpol_cal_ghost	sph_zpl_science_imaging	CALIB	FIBER	IMAGE		ZPL_SCIENCE_IMAGING_REDUCED

5.5.1 Header Keywords Used by ZIMPOL Recipes

"ESO DET CHIP INDEX"
"ESO ZPL STOCK PARAMETER SIGN"
"ESO ZPL STOCK PARAMETER_NAME"
"ESO OCS3 ZIMPOL POL STOKES"
"ESO DET READ CURNAME"
"ESO DET OUT1 X"
"ESO DET OUT1 Y"
"ESO DET BINX" Binning
"ESO DET BINY" Binning
"ESO DET OUT1 NX"
"ESO DET OUT1 NY"
"ESO DET OUT1 OVSCX"
"ESO DET OUT1 OVSCY"
"ESO DET OUT1 PRSCX"
"ESO DET OUT1 PRSCY"
"ESO DET OUT2 X"
"ESO DET OUT2 Y"
"ESO DET OUT2 NX"
"ESO DET OUT2 NY"
"ESO DET OUT2 OVSCX"
"ESO DET OUT2 OVSCY"
"ESO DET OUT2 PRSCX"
"ESO DET OUT2 PRSCY"
"ESO DET BINX"
"ESO DET BINY"
"ESO DET OUT1 X"
"ESO DET OUT1 Y"
"ESO DET OUT1 NX"
"ESO DET OUT1 NY"
"ESO DET OUT1 OVSCX"
"ESO DET OUT1 OVSCY"
"ESO DET OUT1 PRSCX"
"ESO DET OUT1 PRSCY"
"ESO DET OUT2 X"
"ESO DET OUT2 Y"
"ESO DET OUT2 NX"
"ESO DET OUT2 NY"
"ESO DET OUT2 OVSCX"
"ESO DET OUT2 OVSCY"
"ESO DET OUT2 PRSCX"
"ESO DET OUT2 PRSCY"

Chapter 6

Static Calibration Data

All static calibration data for SPHERE can be found the the calibdata subdirectory. The static calibration data for the different three subsystems are located in sphere-0.14.1/spherec/cal The current static calibration data available for SPHERE are the IRDIS and IFS instrument models.

```
> ls -R calibdata calibdata: ifs irdis zimpol
calibdata/ifs: ifs_lenslet_model_Y_H.txt ifs_lenslet_model_Y_J.txt
calibdata/irdis: irdis_instrument_model.txt
calibdata/zimpol:
```

6.1 IFS lenslet model

Several recipes, in particular the wavelength calibration, spectra positions and IFU flat recipes require a model of the lenslet. This model describes how the lenslets are projected onto the detector in some standard dithering position (the “zero” position). In future version this may be extended to include other relevant IFS instrument model parameters, like filter parameters, etc. In the current version, the lenslet model can be provided either as header information in a FITS file or more easily as a simple ASCII text file which is written in the “ini” style format of “KEY = VALUE” pairs on each line.

6.1.1 Parameters of the IFS lenslet model

The IFS default lenslet model is given by the following parameters and values:

The IFS lenslet model may be given using a seperate ASCII file. The standard default model for the J mode would be given in the following way:

Parameter / Keyword	Default Value	Description
HIERARCH ESO DRS IFS DET PIX SIZE	2048	The detector size in pixels
HIERARCH ESO DRS IFS LENS N SIDE	145	The number of lenslets along one side of the BIGRE
HIERARCH ESO DRS IFS LENS SIZE	161.5	The size of lenslets in microns
HIERARCH ESO DRS IFS PIX SIZE	18	The size of detector pixels in microns
HIERARCH ESO DRS IFS SPEC PIX LENGTH	39.0	The model spectra size (in pixels)
HIERARCH ESO DRS IFS SPEC PIX WIDTH	4.93	The model spectra width (in pixels)
HIERARCH ESO DRS IFS ROTANGLE (§)	-11.0	BIGRE rotation angle in degrees (ccw)
HIERARCH ESO DRS IFS BIGRE SCALE	4.5957×10^{-5}	The scale of the BIGRE - this determines the scaling from sky to instrument coordinates in arcsec / microns
HIERARCH ESO DRS IFS BIGRE ROT OFF	-8.7691	BIGRE rotation offset angle in degrees (ccw)
HIERARCH ESO DRS IFS OFF X (§)	8.0	zero point offset in x (in pixels)
HIERARCH ESO DRS IFS OFF Y (§)	2.0	zero point offset in y (in pixels)
HIERARCH ESO DRS IFS SCALE X (§)	1	Spectra pattern scaling in X
HIERARCH ESO DRS IFS SCALE Y (§)	1	Spectra pattern scaling in Y
HIERARCH ESO DRS IFS MAX LAMBDA	1.677 (JH mode) 1.346 (J mode)	Maximum wavelength covered by spectra in microns
HIERARCH ESO DRS IFS MIN LAMBDA	0.951 (JH mode) 0.95 (J mode)	Minimum wavelength covered by spectra in microns
HIERARCH ESO DRS IFS DISPERSON (*)		Dispersion in microns / pixel

Table 6.1: IFS lenslet model parameters. Parameters marked with a (*) are derived from other quanitites and can not be changed directly, parameters mared with a § are fitted for in the spectra positions recipe.

```
[ ESO DRS IFS LENSLET MODEL ]
ESO DRS IFS DET PIX SIZE = 2048
ESO DRS IFS LENS N SIDE = 145
ESO DRS IFS PIX SIZE = 18.000000
ESO DRS IFS LENS SIZE = 161.500000
ESO DRS IFS SPEC PIX LEN = 39.000000
ESO DRS IFS SPEC PIX WIDTH = 4.930000
ESO DRS IFS ROTANGLE = −11.000000
ESO DRS IFS BIGRE SCALE = 0.000045957
ESO DRS IFS BIGRE ROT OFF = −8.769100
ESO DRS IFS OFF X = 2.000000
ESO DRS IFS OFF Y = 8.000000
ESO DRS IFS SCALE X = 1.000000
ESO DRS IFS SCALE Y = 1.000000
ESO DRS IFS MAX LAMBDA = 1.346000
ESO DRS IFS MIN LAMBDA = 0.950000
```

The meaning for the various parameters relating to the spectra (spectra length, minimum and maximum wavelength) are illustrated in Fig.6.1. A cross section of a typical spectrum with a broad band lamp is shown at the top of the figure. Below, the diagram illustrates the meaning of the principal parameters and how the spectrum extraction works. The grey area marks the area as predicted from the lenslet model above, a box with the width and length as specified by the ESO DRS IFS SPEC PIX LEN and ESO DRS IFS SPEC PIX WIDTH parameters. The exact position of the spectrum on the detector is determined by the hexagonal arrangement of the BIGRE lenslet array and the ESO DRS IFS ROTANGLE, ESO DRS IFS ROT OFF, ESO DRS IFS OFF X, ESO DRS IFS OFF Y and ESO DRS IFS SCALE X and ESO DRS IFS SCALE Y parameters. These parameters are fitted for in the spectra positions recipe.

When extracting the spectra, only those pixels are extracted that fall fully inside the predicted spectra model region (the blue area in the diagram). Note that this means that, e.g. for a model spectra length of 39 pixels, the extracted region will always only be 38 pixels long. The minimum and maximum wavelength then refer to the midpoints of the first and last pixels in this model region. The minimum and maximum wavelengths of the model are “guidance” values only: the wavelength calibration recipe for IFS will determine the actual minimum and maximum wavelengths of each spectra region. The dispersion is calculated on the model spectrum and is $\Delta\lambda = (\lambda_{max} - \lambda_{min}) / (L_{model} - 2)$,where $L_{model}$is the model length of the spectra. As seen from the diagram, the length of the extracted spectra $L_{extract} = L_{model} - 1$if L_{model} is an integer value. In that case, the dispersion is: $\Delta\lambda = (\lambda_{max} - \lambda_{min}) / (L_{extract} - 1)$.

6.2 IRDIS Instrument model

Similarly as for IFS, several recipes for IRDIS also rely on an “instrument model”. The IRDIS instrument model is much simpler and currently only contains information on the detector regions corresponding to the different optical paths (left and right paths). It is automatically created in the instrument flat recipe and the model information is stored in the header of the master instrument flat field. It is usually not needed to change this information. The model is valid for the “zero” dithering position. In future version the model may be extended to include other relevant IRDIS instrument model parameters, like filter parameters, etc. In the current version, the lenslet model can be provided most easily as a simple ASCII text file which is written in the “ini” style format of “KEY = VALUE” pairs on each line.

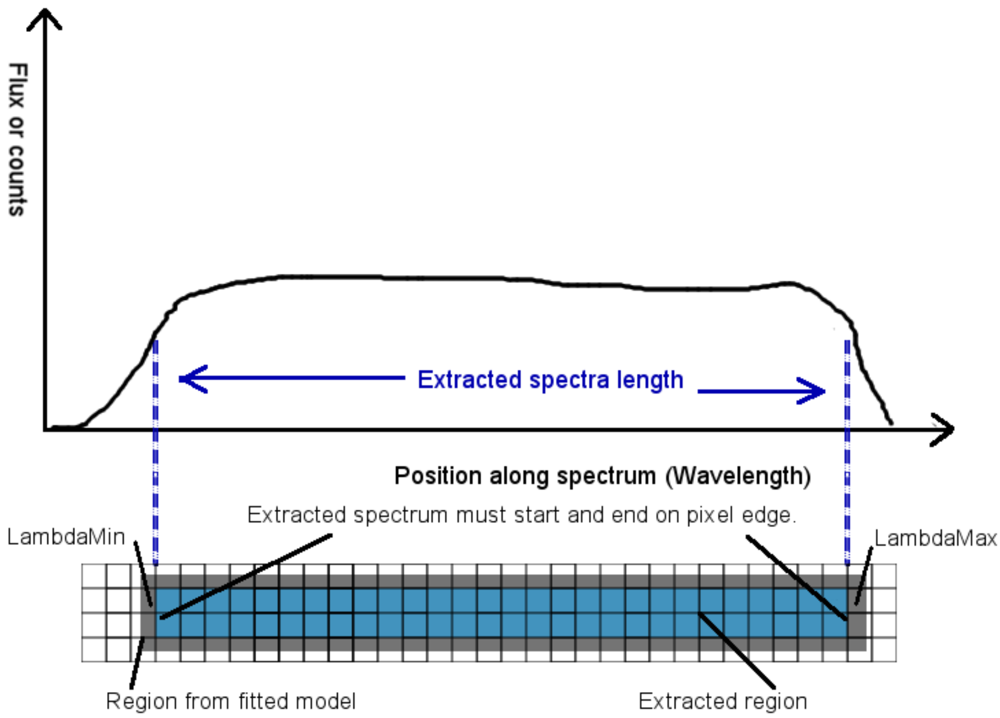


Figure 6.1: The spectra model and spectra extraction for IFS

Chapter 7

Data Reduction Pipeline Data Products Format

7.1 Calibration Products Data Representation

For both IRDIS and IFS, the data reduction pipeline creates calibration products in a variety of formats. The two most general formats are described in this section, but see the description of the individual recipes for more detailed information and information on other data product formats.

7.1.1 The SPHERE "master frame"

The most simple data product produced by the SPHERE pipeline consists of a FITS file with 4 extensions, all containing a single plane (NAXIS = 2) and all having the same number of pixels in both x and y (for IFS and IRDIS this will be NAXIS1 = NAXIS2 = 2048 in most cases). The extensions have the following meaning:

Extension Number	Type	BITPIX	Meaning
1	FLOAT	-32	Image / Main values
2	SHORT	8	Bad or flagged pixels (0 = ok, 1 = bad)
3	FLOAT	-32	Weightmap (e.g. number of pixels that went into result)
4	FLOAT	-32	RMS Error / Other Error Info

7.1.2 Seeing double: The SPHERE double image

For all instruments several calibration (and also raw data) consist of two associated images. This is specifically true for IRDIS, which uses a double optical path close to the detector, and ZIMPOL which stores two interlaced separate images in one readout detector frame using pixel-shifting. Also for IFS some information, like the distortion vector of the lenslet array has an inherit two-component data structure. In nearly all such cases the SPHERE pipelines uses the same data format: a FITS file which consists of a total of 8 extensions. These 8 extensions are:

Extension Number	Type	BITPIX	Meaning
1	FLOAT	-32	Image / Main values for “A” image
2	SHORT	8	Bad or flagged pixels (0 = ok, 1 = bad) for “A” image
3	FLOAT	-32	Weightmap (e.g. number of pixels that went into result) for “A” image
4	FLOAT	-32	RMS Error / Other Error Info for “A” image
5	FLOAT	-32	Image / Main values for “B” image
6	SHORT	8	Bad or flagged pixels (0 = ok, 1 = bad) for “B” image
7	FLOAT	-32	Weightmap (e.g. number of pixels that went into result) for “B” image
8	FLOAT	-32	RMS Error / Other Error Info for “B” image

In this table, the data represented by the “A” and “B” image depend on the specific instrument and recipe: for example, for the sph_ird_science_dpi recipe, the “A” image represents the intensity image, I, and image “B” the polarisation, P.

7.1.3 The SPHERE quad image

Some of ZIMPOL instrument calibration output product consists of 4 associated images. This is the consequence of the fact that one zimpol exposure contains two interlaced images for both phases (0 and PI). Thus, a save quad image FITS file consist of a total of 16 extensions. The 16 extensions are:

Extension Number	Type	BITPIX	Meaning
1	FLOAT	-32	Image / Main values for “A” image (phase 0)
2	SHORT	8	Bad or flagged pixels (0 = ok, 1 = bad) for “A” image (phase 0)
3	FLOAT	-32	Weightmap (e.g. number of pixels that went into result) for “A” image
4	FLOAT	-32	RMS Error / Other Error Info for “A” image (phase 0)
5	FLOAT	-32	Image / Main values for “B” image (phase 0)
6	SHORT	8	Bad or flagged pixels (0 = ok, 1 = bad) for “B” image (phase PI)
7	FLOAT	-32	Weightmap (e.g. number of pixels that went into result) for “B” image(phase 0)
8	FLOAT	-32	RMS Error / Other Error Info for “B” image (phase 0)
9	FLOAT	-32	Image / Main values for “A” image (phase PI)
10	SHORT	8	Bad or flagged pixels (0 = ok, 1 = bad) for “A” image (phase PI)
11	FLOAT	-32	Weightmap (e.g. number of pixels that went into result) for “A” image (phase PI)
12	FLOAT	-32	RMS Error / Other Error Info for “A” image ((phase PI)
13	FLOAT	-32	Image / Main values for “B” image (phase PI)
14	SHORT	8	Bad or flagged pixels (0 = ok, 1 = bad) for “B” image
15	FLOAT	-32	Weightmap (e.g. number of pixels that went into result) for “B” image(phase PI)
16	FLOAT	-32	RMS Error / Other Error Info for “B” image (phase PI)

The quad image format is currently used for the new version of the ZIMPOL master bias and master dark.

Chapter 8

IRDIS Pipeline Recipe Interfaces

8.1 sph_ird_master_dark

Purpose:

Creation of the master dark frame

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_DARK_RAW	Raw data	No	1	Any

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.master_dark.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	master_dark.fits	-
ird.master_dark.save_addprod	bool	Flag to signal whether additional products - in this case the badpixel map - should be saved.	0	-
ird.master_dark.badpixfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products. Only used if badpixel map requested.	static_badpixels.fits	-

Name	Type	Description	Default	Allowed vals.
ird.master_ dark.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.master_ dark.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	0	0-20
ird.master_ dark.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	0	0-20
ird.master_ dark.sigma_ clip	double	Badpixel determination sigma value for clipping.	5.0	0.0-200.0
ird.master_ dark.smoothing	double	The smoothing length to use for calculation of the large scale dark structures. Smoothing is needed for good hotpixel detection.	5.0	0.0-200.0
ird.master_ dark.min_ acceptable	double	The minimum acceptable value. Any pixels with values below this are marked as bad.	-100.0	-
ird.master_ dark.max_ acceptable	double	The maximum acceptable value. Any pixels with values above this are marked as bad.	1000.0	-

Description:

This recipe deals with the creation of the master dark calibration frame.Only raw frames are used in this recipe.The dark is created by combining the input raw frames using the collapsealgorithm specified (usually the clean_mean algorithm).After all input frames are combined in this way, the badpixels are determinedon the result. First a simple thresholding is applied using the parameters min_accepting and max_accepting.A smoothed version of the image is then subtracted to remove large scale variations. The smoothing scale can be changed with the corresponding user parameter.Then sigma clipping is used with the sigma user parameter. All pixels that are further than thespecified sigma value away from the mean are marked as bad in the combined, unsmoothed image..This resulting master dark frame is then written out, A separatehotpixel map is also written out.

Products:

Name	Type	Description
IRD_ MASTER_ DARK	FITS[Im(4)]	The resulting master dark frame. This frame contains 4 different image extensions: the image, badpixels, the weightmap (how many frames contribute to each pixel), and the rms map.
IRD_ STATIC_ BADPIXELMAP	FITS[Im(1)]	An optionally written single extension image of the static badpixels. Note that the content is identical to the second extension in the master dark frame.

8.2 sph_ird_instrument_flat

Purpose:

Determine the instrument flat field

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_FLAT_FIELD_RAW	Raw data	No	1	500
IRD_DARK_RAW	Raw data	Yes	0	500
IRD_STATIC_BADPIXELMAP	Calibration	Yes	0	1
IRD_INSTRUMENT_MODEL	Calibration	Yes	0	1

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.instrument_flat.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	irdis_flat.fits	-
ird.instrument_flat.robust_fit	bool	Controls if fitting method is to be a robust linear fit. This will reduce the effect of cosmic rays and other temporary bad pixels. See e.g. Numerical Recipes for a description of the algorithm	0	-
ird.instrument_flat.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean. This affects only the first processing step, where the illuminated region is determined. It does not affect the actual flat value determination.	2	0,1,2
ird.instrument_flat.clean_mean.reject_high	int	The clean mean reject pixels on high end. This affects only the first processing step, where the illuminated region is determined. It does not affect the actual flat value determination.	0	0-20

Name	Type	Description	Default	Allowed vals.
ird.instrument_flat.clean_mean.reject_low	int	The clean mean reject pixels on low end. This affects only the first processing step, where the illuminated region is determined. It does not affect the actual flat value determination.	0	0-20
ird.instrument_flat.save_addprod	bool	Controls if additional products, in this case a badpixel map should be created.	0	-
ird.instrument_flat.badpixfilename	string	Controls the filename of the badpixel map, if requested for output. Ignored if no make_badpix is FALSE.	instr_flat_badpixels.fits	-
ird.instrument_flat.badpix_lowtolerance	double	The minimum linear threshold value thats acceptable. All pixels in the final flat that have values below this value will be marked as bad.	0.1	-
ird.instrument_flat.badpix_uptolerance	double	The maximum linear threshold value thats acceptable. All pixels in the final flat that have values above this value will be marked as bad.	10.0	-
ird.instrument_flat.badpix_chisqtolerance	double	The maximum error value thats acceptable. All pixels in the final flat that have errors above this value will be marked as bad.	50.0	-
ird.instrument_flat.threshold	double	The thresholding to use to detect illuminated regions. Before the flat is determined all pixels that have counts below a value of the threshold times the mean are masked out. Note that this should only give a very rough masking. It is much preferable to select the regions for flat determination using the static badpixel input frame.	0.1	-

Description:

The instrument flat field recipe for IRDIS is very similar as the detector flatfield recipe for IFS, sph_ifs_detector_flat_field. The flat recipe as described here uses input exposures taken with the narraw band or broad band calibration lamps in any of the IRDIS modes. This flat is used in all subsequent recipes that need to remove the pixel to pixel variation in the signal response of the detector and instrument. It is therefore important thatinput frames are consistently for one particular instrument configuration and thatthe resulting flat is applied only to data taken with matching instrument configurations.As input the recipe requires a series of flat exposures with different median count levels. This may either be achieved by varying the lamp intensity (preferred) or more commonly by varying the exposure time. The recipe also needs as input a series of dark (background) raw frames. Here one such frame is needed for each integration time (DIT) setting present in the input flat frames.For example, if a 10 second exposure flat is included in the input, a corresponding 10 second dark rawframe needs to be given in the input as well. The recipe creates the flats as follows:All raw frames are read in and dark subtracted. The dark subtraction is performed differently than for otherrecipes, and rather than master darks, the recipe actually uses raw dark or background frames. Since thebackground varies significantly depending on the chosen detector integration time, a dark with a matching exposure timeneeds to be subtracted for each flat.If a specific irdis instrument modelis provided via an input frame the irdis instrument model is read from that frame,otherwise a default model is used. This model is used to identifiy the left and rightdetector windows. In the next step, a mask of the illuminated region is created by combining all input exposuresand using a thresholding above the given input threshold value to identify illuminated regionsand masking out non-illuminated regions. Any hot pixels known from the master dark or the provided hotpixel mask are also masked out.The flat fielding procedure descibed below (identical to that for the IFS) is thenapplied to the left and right windows **seperately**.

1. The mean value is determined for the respective window forall exposures.
2. For every pixel $p = (x, y)$, a set of m_i,$v_i(x, y)$ data pairs are stored with m_i being the exposure mean value and $v_i(x, y)$ being the pixel value for exposure i.
3. The flat field value of pixel $p(x, y)$ is defined as the slope $c(x, y)$ of a linear fit F to the data m_i,$v_i(x, y)$.
4. This slope $c(x, y)$ effectively represents the pixel's response to illumination relative to the detector mean response. It *is* the flat field value and comes naturally out of the procedure being close to 1.
5. The fit itself is performed either using a maximum likelyhood methodor a robust fitting method which minimizes the sum of the absolute value of the deviations rather than the sum of the squares of the deviations (see e.g. Numerical Recipes for the algorithm). The robust fitting method will yield better results when significant outliers (e.g. due to cosmic rays) canbe expected.
6. The flat field values are saved as an image as the main product of the recipe.

Aditionally, the recipe may also produce a separate output of all pixels that are identified as non-linear. The criteria for non-linearity are set by the userparameters and can be either pixels that have a flat field value outside specified bounds and/or pixels for which the linear fit produces a reduced chi-squared abovea given threshold value. Note that non-linearity pixel determination is performedon the entire detector region and not the left and right window seperately. For reliable non-linearity flagging using the reduced chi-squared it is necessary to use many high quality input exposures.Since the badpixel treatment is somewhat complicated, some important points: the badpixelsthat are stored in the master flat field itself as produced by this recipe (the second extensionof the main recipe product) contain all the badpixels at this point in the cascade. Pixels thatwere marked as bad from the input static badpixel map are also marked as bad here.The optional static badpixel output that is produced contains strictly only those pixel that theflat field recipe itself deemed to be bad. This does not necessarily include all the badpixelsfrom the static badpixel input file.

Products:

Name	Type	Description
IRD_ FLAT_ FIELD	FITS[Im(4)]	The flat field. This is saved as a FITS file with 4 extensions, the flat values, the badpixels (hotpixels and non-linear pixels), a weight map (number of frames that contributed to each pixel), and the rms

Name	Type	Description
IRD_ STATIC_ BADPIXELMAP	FITS[Im(1)]	Optional output of all the non-linear pixels determined. All pixels as determined in this recipe using the ird.instrument_flat.badpix_ tolerance parameters. This map does NOT include all the dark frame badpixels – it really only includes those badpixels that are bad simply due to the flat field criteria.

8.3 sph_ird_wave_calib

Purpose:

Perform the wavelength calibration

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ WAVECALIB_ RAW	Raw data	No	1	500
IRD_ INSTR_ BG_ RAW	Raw data	Yes	0	1
IRD_ FLAT_ FIELD	Calibration	Yes	0	1
IRD_ MASTER_ DARK	Calibration	Yes	0	1
IRD_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.wave_ calib.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products. ird.	irdis_ wave_ cal.fits	-
ird.wave_ calib.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	1,2
ird.wave_ calib.use_ inskeys	bool	Flag to set whether wavelengths should be set from INS keywords (ignoring then the user command line wavelength parameters).	0	-

Name	Type	Description	Default	Allowed vals.
ird.wave_ calib.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	0	0-20
ird.wave_ calib.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	0	0-20
ird.wave_ calib.threshold	double	Threshold for line detection. This value is used for line detection to determine a rough estimate of dispersion and the line positions before the more careful wavelength calibration is done. The value here should be between the background and the maximal value of the faintest line visible on the image. If the value is negative (default), the threshold is set to ten times the image mean value.	-1.0	-
ird.wave_ calib.smooth	double	When set to a positive value, the raw input data is smoothed with a gauss of the given FWHM before lines are detected and peaks determined.	0.0	-
ird.wave_ calib.wavelength_ line1	double	The wavelength of first line (in microns).	987.72	-
ird.wave_ calib.wavelength_ line2	double	The wavelength of second line (in microns).	1123.71	-
ird.wave_ calib.wavelength_ line3	double	The wavelength of third line (in microns).	1309.0	-
ird.wave_ calib.wavelength_ line4	double	The wavelength of fourth line (in microns).	1545.07	-
ird.wave_ calib.wavelength_ line5	double	The wavelength of fifth line (in microns).	1730.23	-
ird.wave_ calib.wavelength_ line6	double	The wavelength of sixth line (in microns).	2015.33	-
ird.wave_ calib.line_ tolerance	int	The maximal pixel tolerance around which lines are searched for peaks in exposure.	5	-
ird.wave_ calib.number_ lines	int	The number of lines to use. Any input wavelength value for lines with a number higher than the total number of lines to use are ignored.	6	2-6
ird.wave_ calib.degree	int	The polynomial degree to use for the fitting. This should always be at most one less than the number of lines used.	1	1-6
ird.wave_ calib.column_ width	int	The width of the sliding window used to average pixels together before the wavelength solution is found.	179	-
ird.wave_ calib.grism_ mode	bool	Switch to use grism mode (T = grism, F = prism). In grism mode the fitting coefficients c2 = c3 = c4 = 0. The corresponding user parameters are ignored.	0	-
ird.wave_ calib.c2	double	The c2 coefficient in the fit	-43.352	-
ird.wave_ calib.c3	double	The c3 coefficient in the fit	149.723	-

Name	Type	Description	Default	Allowed vals.
ird.wave_ calib.c4	double	The c4 coefficient in the fit	82.442	-

Description:

This recipe performs the wavelength calibration. The raw frames are combined,dark subtracted and flat fielded, flagging any badpixels in the process. After combining the raw frames, the recipewill attempt to detect the lines. For this purpose, the image is sliced intolines parallel to the wavelength direction. For each slice, peaks belongingto the calibration wavelengths are found and assigned to the corresponding input wavelengths. This is done for each calibration wavelength by searching a window region of +/- ird.wave_calib.line_tolerance around the expected pixel for the peak of the calibration wavelength (assuming alinear dispersion and the minimum and maxmimum wavelengths as specifiedin the header of the master instrument flat field frame) for the maximum imagevalue. The actual positions for all input calibration lines are stored and a polynomial fit of input calibration lines versus acutal pixel positions isperformed and used to interpolate all wavelength values between calibration linesfor the image slice. Once all slices inside the spectral region have been processedthe PDT is updated with the new information and written out as the product.

Products:

Name	Type	Description
IRD_ WAVECALIB	FITS[Im(9)]	The wavelength calibration data. This FITS file contains in total six extensions, all containing imaging data. Each image corresponds to one column in the pixel description table (PDT). The order is: wavelength, spectra id, slit id, wavelength width (or error on wavelength), second derivative and illumination fraction. . Additionally saved is the image resulting from a simple combination of all frames, the bad pixel map and an RMS map.

8.4 sph_ird_science_imaging

Purpose:

Science calibration, imaging mode.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_SCIENCE_IMAGING_RAW	Raw data	No	1	500
IRD_MASTER_DARK	Calibration	Yes	0	1
IRD_FLAT_FIELD	Calibration	Yes	0	1
IRD_DISTORTION_MAP	Calibration	Yes	0	1
IRD_STAR_CENTER	Calibration	Yes	0	1
IRD_FCTABLE	Calibration	Yes	0	Any
IRD_STATIC_BADPIXELMAP	Calibration	Yes	0	1
IRD_TFF	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS1 PAC X	double	No	The dithering position in X for the frame in pixels.
ESO INS1 PAC Y	double	No	The dithering position in Y for the frame in pixels.
ESO INS CPRT POSANG	double	Yes	The rotation angle of frame in degrees. Only needed if ADI selected.

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.science_imaging.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	science_imaging.fits	-
ird.science_imaging.outfilename_left	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	science_imaging_left.fits	-
ird.science_imaging.outfilename_right	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	science_imaging_right.fits	-
ird.science_imaging.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median.	1	0,1
ird.science_dbi.keep_fctable	bool	if set to TRUE the recipes internall created field center tables are not deleted.	0	-
ird.science_imaging.use_adi	bool	Flag to control usage of ADI. Currently NOT SUPPORTED ! Use science_imaging instead if you want to use ADI.	0	-
ird.science_imaging.full_frameset_speck	bool	This sets whether speckle frames should be calculated per cube (if set to FALSE) or for the full set of frames (TRUE, default)	1	-
ird.science_imaging.transform_method	int	Transform method to use. 0 is FFT, 1 is CPL_WARP (interpolation).	0	0,1

Name	Type	Description	Default	Allowed vals.
ird.science_ imaging.filter_ method	int	FFT filter method to use. 0 is none, 1 is top hat filter, 2 is Fermi fitler, 3 is Butterworth filter.	0	0,1,2,3
ird.science_ imaging.filter_ rad	double	Radius for FFT top hat and Fermi filters. A non zero value leads to suppression of high frequencies in the fourier domain before framecombination. The value expresses the minimum unsuppressed frequency as fraction of total frequency domain radius (a value of 1 would suppress essentially all frequencies).	0.0	0.0-1.0
ird.science_ imaging.fermi_ temp	double	The temperature parameter for the Fermi filter.	0.0	0.0-1.0
ird.science_ imaging.butter_ pass	double	The pass band frequency for the Butterworth filter, as fraction of total frequency domain radius.	0.0	0.0-1.0
ird.science_ imaging.butter_ stop	double	The stop band frequency for the Butterworth filter, as fraction of total frequency domain radius. This must be larger than the pass frequency.	0.0	0.0-1.0

Description:

This recipe creates the reduced science frames for all science observations with IRDIS in classical imaging mode. The recipesupports dithered frame combination, but does not currently support anyframe de-rotation. Use the science_imaging recipe for cases when de-rotationis needed.The frames are reduced in the following steps:

1. The input raw frames are dark subtracted, if a dark is provided
2. a flat fiels is divided out if it is provided
3. a badpixel map is created for each frame that contains the union of all dark and flat field badpixels
4. the left and right IRDIS subframes are extracted using the IRDIS instrument modelas specified in the header of the flat field (if provided) or the default model otherwise.

Now, for each of the subframes the processing is as follows:

1. high frequency filtering. If the filter radius f_r is set to a value larger than 0, a tophat frequency filter is applied, masking out all frequencies above the value of $f > f_r \times f_{max}$,where f_{max} is the maximum frequency in the FFT. For noise filterings a value between 0.9 and 0.99 aregood values to use for f_r.
2. FFT shifting of image to recenter the image
3. Application of the distortion map to image
4. Shifting and distortion map application to badpixel map using geometrical approach

All these processed frames are then saved as temporary files. Then these frames are combined using theselected combination algorithm.

- if a weighted mean is selected, a weightmap is calculated first taking the median frame as a reference frame andweighing down the other frames depending on the difference in values. Note that this is still a very experimental option andit is still to be defined what weighting sheme would be optimal. All badpixels get assigned the weight of 0. Frames are thencombined taking the individual weights into account.
- if a mean is selected, frames are combined using a mean, after first rejecting all bad pixels.
- if a median is selected, frames are combined by taking the median pixel value at each pixel position. This procedureignores the badpixels.

The obtained combined results for the left and right IRDIS field of view are saved in a single FITS file with 8 extensions, followingthe layout for a double master frame: the first four extensions being the image, badpixelmap, N map, and rms for the left field andthe second set of four extensions being the equivalent for the right field.

Products:

Name	Type	Description
IRD_ SCIENCE_ IMAGING	FITS[Im(4)]	The main science frame. The FITS file contains 4 extensions: the image, the badpixels, the rms error and a weightmap. All show the whole detector.

8.5 sph_ird_science_dbi

Purpose:

Science calibration, DBI mode.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ SCIENCE_ DBI_ RAW	Raw data	No	1	500
IRD_ MASTER_ DARK	Calibration	No	1	1
IRD_ FLAT_ FIELD	Calibration	No	1	1
IRD_ DISTORTION_ MAP	Calibration	Yes	0	1
IRD_ FILTER_ TABLE	Calibration	Yes	0	1
IRD_ STAR_ CENTER	Calibration	Yes	0	1
IRD_ FCTABLE	Calibration	Yes	0	Any
IRD_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS IRD DUAL FILTER LAMBDA LEFT	double	Yes	The central wavelength of the filter on left. Only needed if SDI requested.
ESO DRS IRD DUAL FILTER LAMBDA RIGHT	double	Yes	The central wavelength of the filter on right. Only needed if SDI requested.

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.science_ dbi.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	science_ dbi.fits	-
ird.science_ dbi.save_ addprod	bool	Flag signalling whether additional products should be saved. these are the individual, adi combined when reuired, products for the left and right fields Flag signalling whether intermediate products should be saved/kept on disk. These are the prime starting points for independent differential analyses with third-party software! Not that this switch is currently ignored and the intermediate products are always kept on disk!	0	-
ird.science_ dbi.outfilename_ left	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	science_ dbi_ left.fits	-
ird.science_ dbi.make_ template	bool	if set to TRUE the recipe creates an empty template of the field center table to be filled by hand.	0	-
ird.science_ dbi.keep_ fctable	bool	if set to TRUE the recipes internall created field center tables are not deleted.	0	-
ird.science_ dbi.outfilename_ right	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	science_ dbi_ right.fits	-
ird.science_ dbi.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median.	1	0,1
ird.science_ dbi.use_ adi	bool	Flag to control usage of ADI	0	-
ird.science_ dbi.use_ sdi	bool	Flag to control usage of SDI	0	-
ird.science_ dbi.minr	double	The minimum radius of the annulus used to renormalise the flux for SDI.	4.0	0.0-512.0
ird.science_ dbi.maxr	double	The maximum radius of the annulus used to renormalise the flux for SDI.	40.0	0.0-512.0

Name	Type	Description	Default	Allowed vals.
ird.science_dbi.full_frameset_speck	bool	This sets whether speckle frames should be calculated per cube (if set to FALSE) or for the full set of frames (TRUE, default)	1	-
ird.science_dbi.transform_method	int	Transform method to use. 0 is FFT, 1 is CPL_WARP (interpolation).	0	0,1
ird.science_dbi.filter_method	int	FFT filter method to use. 0 is none, 1 is top hat filter, 2 is Fermi fitler, 3 is Butterworth filter.	0	0,1,2,3
ird.science_dbi.filter_rad	double	Radius for FFT top hat and Fermi filters. A non zero value leads to suppression of high frequencies in the fourier domain before framecombination. The value expresses the minimum unsuppressed frequency as fraction of total frequency domain radius (a value of 1 would suppress essentially all frequencies).	0.0	0.0-1.0
ird.science_dbi.fermi_temp	double	The temperature parameter for the Fermi filter.	0.0	0.0-1.0
ird.science_dbi.butter_pass	double	The pass band frequency for the Butterworth filter, as fraction of total frequency domain radius.	0.0	0.0-1.0
ird.science_dbi.butter_stop	double	The stop band frequency for the Butterworth filter, as fraction of total frequency domain radius. This must be larger than the pass frequency.	0.0	0.0-1.0
ird.science_dbi.window_size	int	When set to a non zero value, the recipe uses a special subwindow mode, where only cut-outs are of the given size are used (the cut out is made after dark and flat have been applied and the subfields have been extracted). For example to use only the central 128 pixels for both left and right subfields use window_size=128.	0	0-1024
ird.science_dbi.star_r	double		2.0	-
ird.science_dbi.bg_r1	double		2.0	-
ird.science_dbi.bg_r2	double		3.0	-

Description:

This recipe creates the reduced science frames for all science observations with IRDIS in DBI maging mode. The recipesupports dithered frame combination, as well as ADI and SDIThe frames are reduced in the following steps:

1. The input raw frames are dark subtracted, if a dark is provided
2. a flat field is divided out if it is provided
3. a badpixel map is created for each frame that contains the union of all dark and flat field badpixels

4. the left and right IRDIS subframes are extracted using the IRDIS instrument modelas specified in the header of the flat field (if provided) or the default model otherwise.

Now, for each of the subframes the processing is as follows:

1. high frequency filtering. If the filter radius f_r is set to a value larger than 0, a tophat frequency filter is applied, masking out all frequencies above the value of $f > f_r \times f_{max}$,where f_{max} is the maximum frequency in the FFT. For noise filterings a value between 0.9 and 0.99 aregood values to use for f_r.
2. FFT or warp shifting of image to recenter the image
3. Application of the distortion map to image
4. Shifting and distortion map application to badpixel map using geometrical approach

All these processed frames are then saved as temporary files. These are then combined to create a reference speckle image.Now the ADI or SDI steps are performed if one of them or both are selected. If not is selected, these steps are skipped.

1. The speckle frame is subtracted.
2. if SDI is selected, scaling of the images using FFT around the image center – the angle should be given as part of thefield center table provided in the input of the recipe.
3. if ADI is selected, rotation of the images using FFT around the image center – the angle should be given as part of thefield center table provided in the input of the recipe.

Then these frames are combined using theselected combination algorithm.

- if a weighted mean is selected, a weightmap is calculated first taking the median frame as a reference frame andweighing down the other frames depending on the difference in values. Note that this is still a very experimental option andit is still to be defined what weighting sheme would be optimal. All badpixels get assigned the weight of 0. Frames are thencombined taking the individual weights into account.
- if a mean is selected, frames are combined using a mean, after first rejecting all bad pixels.
- if a median is selected, frames are combined by taking the median pixel value at each pixel position. This procedureignores the badpixels.

The obtained combined results for the left and right IRDIS field of view are saved in a single FITS file with 8 extensions, followingthe layout for a double master frame: the first four extensions being the image, badpixelmap, N map, and rms for the left field andthe second set of four extensions being the equivalent for the right field.Some additional notes:

- The static badpixel frame is optional and the badpixels defined there will be combined (using a logical OR) with badpixels inthe dark or flat.
- Before the images are transformed (rotated and/or shifted) badpixels are interpolated. Interpolation happens irrespective of algorithm choice.The interpolation is a simple 8 neighbbour pixel average. Incase a number of n $<$ 8 neighbour pixels are also bad, 8 - n values areused. In case all neighbour pixels are bad, the interpolation simply copies the value from nearest non bad pixel.
- While a filter table is not strictly required, no scaling will be done if SDI is selected, leading to zero images.

Products:

Name	Type	Description
IRD_SCIENCE_DBI	FITS[Im(4)]	The main science frame. The FITS file contains 4 extensions: the image, the badpixels, the rms error and a weightmap for the total or difference of the left and right fields of view. Optionally, left and right field can be produced as additioanl products

8.6 sph_ird_science_dpi

Purpose:

Science calibration, DPI mode.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_SCIENCE_DPI_RAW	Raw data	No	1	Any
IRD_MASTER_DARK	Calibration	No	1	1
IRD_FLAT_FIELD	Calibration	No	1	1
IRD_DISTORTION_MAP	Calibration	Yes	0	1
IRD_STAR_CENTER	Calibration	Yes	0	1
IRD_FCTABLE	Calibration	Yes	0	Any
IRD_STATIC_BADPIXELMAP	Calibration	Yes	0	1

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.science_dpi.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	science_dpi.fits	-
ird.science_dpi.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median.	1	0,1
ird.science_dpi.use_adi	bool	Flag to control usage of ADI	0	-

Name	Type	Description	Default	Allowed vals.
ird.science_ dpi.use_ sdi	bool	Flag to control usage of SDI	0	-
ird.science_ dpi.keep_ fctable	bool	if set to TRUE the recipes internall created field center tables are not deleted.	0	-
ird.science_ dpi.minr	double	The minimum radius of the annulus used to renormalise the flux for SDI.	4.0	0.0-512.0
ird.science_ dpi.maxr	double	The maximum radius of the annulus used to renormalise the flux for SDI.	40.0	0.0-512.0
ird.science_ dpi.full_ frameset_ speck	bool	This sets whether speckle frames should be calculated per cube (if set to FALSE) or for the full set of frames (TRUE, default)	1	-
ird.science_ dpi.transform_ method	int	Transform method to use. 0 is FFT, 1 is CPL_WARP (interpolation).	0	0,1
ird.science_ dpi.filter_ method	int	FFT filter method to use. 0 is none, 1 is top hat filter, 2 is Fermi fitler, 3 is Butterworth filter.	0	0,1,2,3
ird.science_ dpi.filter_ rad	double	Radius for FFT top hat and Fermi filters. A non zero value leads to suppression of high frequencies in the fourier domain before framecombination. The value expresses the minimum unsuppressed frequency as fraction of total frequency domain radius (a value of 1 would suppress essentially all frequencies).	0.0	0.0-1.0
ird.science_ dpi.fermi_ temp	double	The temperature parameter for the Fermi filter.	0.0	0.0-1.0
ird.science_ dpi.butter_ pass	double	The pass band frequency for the Butterworth filter, as fraction of total frequency domain radius.	0.0	0.0-1.0
ird.science_ dpi.butter_ stop	double	The stop band frequency for the Butterworth filter, as fraction of total frequency domain radius. This must be larger than the pass frequency.	0.0	0.0-1.0

Description:

This recipe creates the reduced science frames for all science observations with IRDIS in DPI mode. The recipe isessentially identical to the DBI science recipe, except that the final result is saved asas I and P frames (rather than left and right field of views).Please see the description of the IRDIS DBI recipe for more details on the processing.

Products:

Name	Type	Description

Name	Type	Description
IRD_SCIENCE_DPI	FITS[Im(8)]	The main science frame. The FITS file contains 8 extensions: the first 4 extensions contain the image, the badpixels, the rms error and a weightmap of the polarisation. The last 4 extensions contain the same information for the intensity image.

8.7 sph_ird_science_spectroscopy

Purpose:

Science calibration, spectroscopy mode.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_SCIENCE_SPECTROSCOPY_RAW	Raw data	No	1	500
IRD_MASTER_DARK	Calibration	No	1	1
IRD_FLAT_FIELD	Calibration	No	1	1
IRD_ATMOSPHERIC	Calibration	Yes	0	1
IRD_STATIC_BADPIXELMAP	Calibration	Yes	0	1

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.science_spectroscopy.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	science_spectroscopy.fits	-
ird.science_spectroscopy.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median.	0	0,1

Description:

This recipe creates the actual science frames for spectroscopy mode.In spectroscopy mode, frames are not dithered and so this recipe performs a simple processing of dark subtraction and flat fielding beforeusing a user specified method to combine the frames. If a atmospheric calibration is provided this is subtracted from the result.

Known Issues:

The recipe is not using the wavelength calibration file. In order to remove the effects of the wavelength dependence of the flat, the recipe should really use a series of flats taken at different wavelengths and construct a 'super' flat from this using the wavelength calibration file, in the same way as this is done for IFS. However, it is not clear if this is in fact required or if the wavelength dependence of the flat has too small an effect on the spectroscopic data reduction to make it necessary to perfom this wavelength dependent calibration.

Products:

Name	Type	Description
IRD_ SCIENCE_ SPECTROSCOPY_ LEFT	FITS[Im(4)]	The reduced spectroscopy frame of the left FOV. This frame contains a full detector image with the 2D spectrum. The format is FITS with 4 extensions, the actual image data, the badpixels, the ncombmap (how many frames contributed to each pixel), and a RMS map.
IRD_ SCIENCE_ SPECTROSCOPY_ RIGHT	FITS[Im(4)]	Same as the above, but for the right FOV

8.8 sph_ird_astrometry

Purpose:

Measure plate scale and angle

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ ASTROMETRY_ RAW	Raw data	No	1	500
IRD_ MASTER_ DARK	Calibration	No	1	1
IRD_ FLAT_ FIELD	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS IFS CORONO	string	No	The keyword that specified if the coronagraph is in or out.
ESO DRS IRD DUAL FILTER LAMBDA LEFT	double	Yes	The central wavelength of the filter on left. Only needed if SDI requested.
ESO DRS IRD DUAL FILTER LAMBDA RIGHT	double	Yes	The central wavelength of the filter on right. Only needed if SDI requested.
ESO INS1 PAC X	double	No	The dithering position in X for the frame in pixels.
ESO INS1 PAC Y	double	No	The dithering position in Y for the frame in pixels.
ESO INS CPRT POSANG	double	Yes	The rotation angle of frame in degrees. Only needed if ADI selected.

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.astrometry.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	astrometry.fits	-
ird.astrometry.outfilename_left	string	This option is used only for debugging purposes. Please do not use.	astrometry_left.fits	-
ird.astrometry.outfilename_right	string	This option is used only for debugging purposes. Please do not use.	astrometry_right.fits	-
ird.astrometry.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.astrometry.clean_mean.reject_high	int	The clean mean reject pixels on high end.	1	0-20
ird.astrometry.clean_mean.reject_low	int	The clean mean reject pixels on low end.	1	0-20
ird.astrometry.use_adi	bool	Set to TRUE if input frames are to be de-rotated and ADI should be applied.	0	-
ird.astrometry.use_sdi	bool	Set to TRUE if input frames are to be rescaled and SDI should be applied.	0	-
ird.astrometry.filter_radius	double	Filter radius for ADI and SDI, below which to suppress frequencies.	0.0	0.0-1.0
ird.astrometry.companion_x	double	The x position of companion relative to parent star in milliarcsec.	0.0	-550.0-550.0
ird.astrometry.companion_y	double	The y position of companion relative to parent star in milliarcsec.	0.0	-550.0-550.0

Description:

This recipe creates astrometry information. It does this by first reducingthe input raw frames, which should be of a binary star system, in the standard way. The recipe then attempts to detect both the star and thecompanion. From the relative position between star and companion therecipe determines the pixel scale and the rotation angle wrt. the northdirection. The input user parameter of star to companion relative position (in milliarcsec) as well as the instrument rotation angle keywords as set by the instrument control software (in the raw frames) are used forthis purpose. Note that contrary to the science_dbi and science_imagingrecipe this recipe outputs as image data a combinaed left and rightopitcal path image.

Products:

Name	Type	Description
IRD_ ASTROMETRY	FITS[Im(4)]	The resulting reduced science frame. Contains the astrometry keywords for plate scale and rotation angle to north in header. The FITS file contains image, badpixels, rms and weightmap for a combination image of both optical paths.

8.9 sph_ird_star_center

Purpose:

Determine the field centre

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ STAR_ CENTER_ WAFFLE_ RAW	Raw data	No	1	Any
IRD_ MASTER_ DARK	Calibration	Yes	0	1
IRD_ FLAT_ FIELD	Calibration	Yes	0	1
IRD_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.star_ center.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	star_ center.fits	-
ird.star_ center.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.star_ center.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	1	0-20
ird.star_ center.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	1	0-20
ird.star_ center.sigma	double	The sigma threshold to use for source detection	10.0	-

Name	Type	Description	Default	Allowed vals.
ird.star_ center.use_ waffle	bool	Flag to whether to expect a waffle image (4 images in cross formation) or not (single central fit).	1	-
ird.star_ center.qc	bool	If set QC output for this recipe is produced.	0	-
ird.star_ center.unsharp_ window	int	Before finding centres an unsharp algorithm is used on the image. This specifies the window width for the mask in pixels.	4	-

Description:

This recipe creates a table with centre star positions. The input raw framesare each reduced by subtracting the dark and applying the flat provided.After sorting the frames, the recipe only reduces the image data of the waffle images.An optional mask frame may be given, of the same dimenstions as the raw input frames,which allows masking out of regions before the point sources are detected. This can mainly be used on images where despite use of a coronagraph a significant central signal is present.The left and right parts of the illuminated detector regions are extractedand left and right part are seperately analysed using a aperture detectionalgorithm. The aperture detection algorithm detects all connected regionsof at least 4 pixels size (area) that are the given sigma above the background.The so detected waffle stars are then used to contruct a geometric centre of all stars found. This is then the frame centre.The recipe also works for the case that there is only one star (e.g. the choronograph is out and no waffle stars are formed).After frame centers have been determined for all waffle images an internaltable is created with an entry for each waffle image, giving the time of thestart of the exposure and the centre information.The recipe reads the position of the IRDIS DMS from the header of the raw frames, divides by 18.0 to convert from micron to pixels, and stores them in the output table.

Known Issues:

While this recipe is functional, its requirements are fully settled. The recipe implementes the current baseline of how star centering is foreseen in IRDIS.

Products:

Name	Type	Description
IRD_ STAR_ CENTER	FITS[Table]	The table of stellar center positions as a FITS table, with one row for each input raw frame. The order is the same as the order of input raw frames.

8.10 sph_ird_atmospheric

Purpose:

Measure the atmospheric effect

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ ATMOSPHERIC_ RAW	Raw data	No	1	500
IRD_ MASTER_ DARK	Calibration	No	1	1
IRD_ FLAT_ FIELD	Calibration	No	1	1
IRD_ WAVECALIB	Calibration	Yes	0	1

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.atmospheric.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ird_ atmospheric.fits	-
ird.atmospheric.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 3 = Weighted mean.	0	0,1
ird.atmospheric.airmass	double	The airmass that for the raw frames. If the value is >= 1, this value is written to the header of the product, rather than the value taken from the first raw input file.	0.0	0.0-200.0

Description:

This recipe is responsible for creating sky spectra. Its identical to the science_spectroscopy recipe.

Products:

Name	Type	Description
IRD_ ATMOSPHERIC	FITS[Im(4)]	The frame containing the calculated, estimated absorption spectrum of the atmosphere.

8.11 sph_ird_flux_calib

Purpose:

Calibrate the effect of coronagraph

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_FLUX_CALIB_CORO_RAW	Raw data	No	1	100
IRD_FLUX_CALIB_NO_CORO_RAW	Raw data	No	1	100
IRD_MASTER_DARK	Calibration	No	1	1
IRD_FLAT_FIELD	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS IFS CORONO	string	No	The keyword that specified if the coronagraph is in or out.

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.flux_calib.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	flux_calib.fits	-
ird.flux_calib.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.flux_calib.clean_mean.reject_high	int	The clean mean reject pixels on high end.	1	0-20
ird.flux_calib.clean_mean.reject_low	int	The clean mean reject pixels on low end.	1	0-20

Description:

This recipe calibrates the effect of the coronagraph on the detected number ofcounts. For this purpose the raw frames with and without coronagraph are reducedseperately in the standard way (dark subtraction, flat fielding). The recipe then measures the total flux in the coronagraph and the non-coronograph framesand saves the ratio as a keyword together with the reduced images.

Products:

Name	Type	Description
IRD_FLUX_CALIB	FITS[Im(4)]	The reduced frame with the calibration keywords in header.

8.12 sph_ird_ins_throughput

Purpose:

Measure the instrument induced polarisation

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_INS_THROUGHPUT_RAW	Raw data	No	1	500
IRD_MASTER_DARK	Calibration	No	1	1
IRD_FLAT_FIELD	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS1 PAC X	double	No	The dithering position in X for the frame in pixels.
ESO INS1 PAC Y	double	No	The dithering position in Y for the frame in pixels.

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.ins_throughput.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ins_throughput.fits	-
ird.ins_throughput.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.ins_throughput.clean_mean.reject_high	int	The clean mean reject pixels on high end.	1	0-20
ird.ins_throughput.clean_mean.reject_low	int	The clean mean reject pixels on low end.	1	0-20
ird.ins_throughput.flux	double	The input flux of the observed source in 10^-10 erg cm^-2 s^-1 A^-1.	0.0	0.0-30.0

Description:

This recipe measure the instrument throughput. The input raw frames are reducedjust as for the science recipes (see e.g. sph_ird_science_imaging), subtractingthe dark and applying the flat field. The product contains information on the total flux and throughput, as calculated from the total measured flux and the user input flux.

Products:

Name	Type	Description
IRD_ INS_ THROUGHPUT	FITS[Im(4)]	The reduced throughput frame with throughput keywords in header. The file has 4 extensions, the image itself, the badpixel map, the rms error map and a weightmap.

8.13 sph_ird_sky_bg

Purpose:

Determine the sky background

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ SKY_ BG_ RAW	Raw data	No	1	500
IRD_ MASTER_ DARK	Calibration	No	1	1
IRD_ FLAT_ FIELD	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS1 PAC X	double	No	The dithering position in X for the frame in pixels.
ESO INS1 PAC Y	double	No	The dithering position in Y for the frame in pixels.

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.sky_ bg.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	sky_ bg.fits	-
ird.sky_ bg.lsf_ outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	sky_ bg_ fit.fits	-
ird.sky_ bg.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 3 = Weighted mean.	1	0,1,3
ird.sky_ bg.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	1	0-20
ird.sky_ bg.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	1	0-20

Name	Type	Description	Default	Allowed vals.
ird.sky_ bg.fitorder	int	The fitting order to use for the 2D polynomial fit of the background.	2	1-7

Description:

This recipe creates the sky background. It reduced the raw frames in thesame way as the sph_ird_science_* recipes by dark subtracted and flat fieldingthe frames before combining them taking any dithering into account (no framescaling or de-rotation is performed). The resulting frame is the combinationof the left and right optical paths (so only has a 1k by 1k size). Note thatthis means that the sky bacgkround should not use raw frames which have filtersthat are different for the left and right optical paths.The sky background frame is saved as the main product along with a 2D polynomialfit of the sky background.

Products:

Name	Type	Description
IRD_ SKY_ BG	FITS[Im(4)]	The main background frame (unsmoothed). The FITS file contains 4 extensions: the image, the badpixels, the rms error and a weightmap. All are 1024x1024 pixels and are based on the TOTAL of frames and left and right fields!
IRD_ SKY_ BG_ FIT	FITS[Im(4)]	The smoothed frame (2D polynomial fit) of the background. The FITS file contains 4 extensions: the image, the badpixels, the rms error and a weightmap. All are 1024x1024 pixels and are based on the TOTAL of frames and left and right fields!

8.14 sph_ird_tff

Purpose:

Measures large scale telescope flat field

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_TFF_RAW	Raw data	No	1	500
IRD_MASTER_DARK	Calibration	No	1	1
IRD_FLAT_FIELD	Calibration	No	1	1
IRD_STATIC_BADPIXELMAP	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS1 PAC X	double	No	The dithering position in X for the frame in pixels.
ESO INS1 PAC Y	double	No	The dithering position in Y for the frame in pixels.

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.tff.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products. ird.	telescope_flat.fits	-
ird.tff.lsf_outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products. ird.	large_scale_flat.fits	-
ird.tff.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median.	1	0,1
ird.tff.clean_mean.reject_high	int	The clean mean reject pixels on high end.	1	0-20
ird.tff.clean_mean.reject_low	int	The clean mean reject pixels on low end.	1	0-20
ird.tff.fitorder	int	The fitting order to use for the 2D polynomial fitting.	2	1-7
ird.tff.robust_fit	bool	Controls if fitting method is to be a robust linear fit. This will reduce the effect of cosmic rays and other temporary bad pixels. See e.g. Numerical Recipes for a description of the algorithm	0	-
ird.tff.badpix_lowtolerance	double	The minimum linear threshold value thats acceptable. All pixels in the final flat that have values below this value will be marked as bad.	0.1	-
ird.tff.badpix_uptolerance	double	The maximum linear threshold value thats acceptable. All pixels in the final flat that have values above this value will be marked as bad.	10.0	-
ird.tff.badpix_chisqtolerance	double	The maximum error value thats acceptable. All pixels in the final flat that have errors above this value will be marked as bad.	50.0	-

Description:

This recipe creates a large scale flat field of the entire system of telescope plus instrument. The flat field is created from on-sky flats.The reduction is identical to that for the science recipes. Dithering is allowed but not taken into account in the reduction and no rescaling or de-rotation is performed. When the flat has been created, a smoothed version is made which is created by fitting theflat with a 2D polynomial of the specified fitting order. The smoothed flat field obtained in this way is saved.

Products:

Name	Type	Description
IRD_ TFF	FITS[Im(5)]	The telescope flat field. This frame contains 4 different image extensions: the image, badpixels, the weightmap (how many frames contribute to each pixel), the rms map, and an additional map of linear bad pixels
IRD_ LSF	FITS[Im(4)]	The large scale flat field. A version of the above smoothed by fitting a 2D polynomial. This frame contains 4 different image extensions: the image, badpixels, the weightmap (how many frames contribute to each pixel), and the rms map.

8.15 sph_ird_psf_reference

Purpose:

Creation of a PSF reference image

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ PSF_ REFERENCE_ RAW	Raw data	No	1	500
IRD_ MASTER_ DARK	Calibration	No	1	1
IRD_ FLAT_ FIELD	Calibration	No	1	1
IRD_ DISTORTION_ MAP	Calibration	Yes	0	1
IRD_ STAR_ CENTER	Calibration	Yes	0	1
IRD_ FCTABLE	Calibration	Yes	0	Any
IRD_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS1 PAC X	double	No	The dithering position in X for the frame in pixels.
ESO INS1 PAC Y	double	No	The dithering position in Y for the frame in pixels.

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.psf_ reference.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	psf_ reference.fits	-
ird.psf_ reference.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	1	0,1
ird.psf_ reference.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	1	0-20
ird.psf_ reference.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	1	0-20
ird.science_ imaging.filter_ radius	double	Filter radius for ADI framecombination. A non zero value leads to suppression of high frequencies in the fourier domain before framecombination. The value expresses the minimum unsuppressed frequency as fraction of total frequency domain radius (a value of 1 would suppress essentially all frequencies).	0.0	0.0-1.0

Description:

This recipe is a specific purpose science reduction of a PSF reference image.The recipe is identical to the sph_ird_science_imaging/dpi recipes.

Products:

Name	Type	Description
IRD_ PSF_ REFERENCE	FITS[Im(4)]	The reduced PSF image. Saved as a FITS file with 4 extensions containing image, badpixels, rms and weightmap.

8.16 sph_ird_pol_zpa_eff

Purpose:

Measure polarisation zero point.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_POL_ZPA_EFF_RAW	Raw data	No	1	500
IRD_MASTER_DARK	Calibration	No	1	1
IRD_FLAT_FIELD	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS1 PAC X	double	No	The dithering position in X for the frame in pixels.
ESO INS1 PAC Y	double	No	The dithering position in Y for the frame in pixels.
ESO DRS IRD HWP2 SETTING	double	No	The setting of the HWP2 as an angle expressed in degrees.

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.pol_zpa_eff.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	pol_zpa_eff.fits	-
ird.pol_zpa_eff.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.pol_zpa_eff.clean_mean.reject_high	int	The clean mean reject pixels on high end.	1	0-20
ird.pol_zpa_eff.clean_mean.reject_low	int	The clean mean reject pixels on low end.	1	0-20

Description:

This algorithm performs measurement of the polarization zero point angle. For this purpose, the raw frames should be taken with the same azimuthal and elevation.The frames are dark subtracted and flat fielded. The sub-images A and B are then extracted to give the measures of I_+ and I_|| values on the detector for each raw frame. The frames with the same IR_HWP and de-rotator setting are added, creating for each polarization (Q and U) a master frame. The zero point for each de-rotator setting dr is then ZP=0.5*arctan[(U/I)/(Q/I)]. The zero point is calculated for all pixels and then averaged over all pixels and the rms determined for QC purposes.

Products:

Name	Type	Description

Name	Type	Description
IRD_ POL_ ZPA_ EFF	FITS[Im(4)]	The polarization zero point measurements. This product is saved as a FITS file with values, badpixels, rms and weightmap. The result shows the zero point angle as defined above.

8.17 sph_ ird_ tel_ pol_ offset

Purpose:

Measures telescope induced polarisation

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ TEL_ POL_ RAW	Raw data	No	1	100
IRD_ MASTER_ DARK	Calibration	No	1	1
IRD_ FLAT_ FIELD	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS1 PAC X	double	No	The dithering position in X for the frame in pixels.
ESO INS1 PAC Y	double	No	The dithering position in Y for the frame in pixels.

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.tel_ pol_ offset.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products. ird.	tel_ pol_ offset.fits	-
ird.tel_ pol_ offset.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.tel_ pol_ offset.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	1	0-20
ird.tel_ pol_ offset.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	1	0-20

Description:

This recipe is for all practical pruposes identical to the ird science DPI recipe.

Products:

Name	Type	Description
IRD_ TEL_ POL	FITS[Im(8)]	The telescope induced polarisation. The file is a FITS file with in total 8 extensions. The first 4 extensions contain image, badpixelmap, rms and weightmap for the intensity I, the last 4 extensions the same information in the same order for the polarization P.

8.18 sph_ird_gain

Purpose:

Measure the detector gain

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ GAIN_ RAW	Raw data	No	4	Any
IRD_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.gain.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ird_ gain_ map.fits	-
ird.gain.nonlin_ filename	string	The output filename for the non-linearity map. Please also see the esorex documentation	ird_ nonlin_ map.fits	-

Name	Type	Description	Default	Allowed vals.
ird.gain.nonlin_ bpixname	string	The output filename for the non linear bad pixel map. Please also see the esorex documentation for naming of output products.	ird_ nonlin_ bpix.fits	-
ird.gain.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.gain.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	0	0-20
ird.gain.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	0	0-20
ird.gain.order	int	The fitting order to use, can be 1 (for linear only) or 2 (for lin+quadratic).	2	1-2
ird.gain.lin_ tolerance	double	The allowed maximum absolute value of the second order of the polynomial fit. Any pixels that have an absolute value for the second order polynomial coefficient above this value are considered non-linear and marked as bad in the non-linearity map	100.0	-
ird.gain.preproc	bool	If set to TRUE, the raw frames are first processed to remove any offset trends within data cubes	1	-
ird.gain.vacca	bool	Choose the special noise calculation by Vacca et al. (2004) that takes the number of fowler samples into account.	0	-

Description:

The gain recipe calculates the gain for the detector and derives a mask of nonlinear pixels. The input is assumed to be a series of data cubes, each containg a single extension with N > 3 planes that each contain a single exposure. The mean count for each input cube shoulld be differenteither by increasing the intensity of the illumination source or by using different exposuretimes. Note that in the latter case the recipe only produces the correct output if the detector gain is independent of the read out mode.The gain recipe, as well as the ron recipe, have a special optional preprocessing step, whichcorrects some possible bias due to readout electronics settings by first subtracting the median of each input cube from each image in the cube.The gain recipe offers to algorithms to calculate the gain, one straightforward fitting algorithm and a morecomplex fitting algorithm that takes the correct number of fowler samples into account. The second algorithmis switched on using the vacca user parameter and ispreferable for accurate gain determinations but can currently not be used to calcuate the detector non-linearity. It is therefore recommended to set the user parameter vacca to 1 whenan accurate gain measurement is needed but not non-linearity measurement is needed and 0 in all other cases.In particular for pure monitoring purposes to discover trends in the gain the simpler algorithm is sufficient. For both algorithms the general procedure is similar:The recipes calculates the gain by first collapsing allinput cubes to create a single mean image and variance image. The collapse algorithm specified(clean mean by default) and algorithm parameters are used for this process. Once a mean and varianceimage has been determined the median of the mean image and the corresponding variance is takenas one data point. The collection of input cubes then lead to a collection of data points of median andvariance, giving measurements of the variance vs. median relation for the detector. This is then fittedusing a polynomial of the specified order (usually 1 or 2). The slope of this curve is the inverseof the gain while the offset gives an estimate of the read out noise. Note that the read out noiseestimate obtained here may not be accurate. Please use the dedicated ron recipe to obtain a more accurate estimate of the RON. The estimates of gain and ronare written as keywords in the main recipe product FITS file.If the vacca parameter is set, the recipe corrects the fitting coefficients for the different noiseproperties expected for different fowler samples. For example, for double correlated reads this corrects the ron by a factor of 2.If the vacca parameter is not set. the recipe determines non linear pixels in a second step. This is done by performing the gain fitting procedure above for each individual pixel. The resulting map of the gain is the data in the first extension of the main productFITS file. Note that the pixel-by-pixel gain values are often very noisy and can not be used to obtainprecise gain measurements. Many exposures per input cube are needed to perform accurate pixel fitting.If the fitorder specified is larger than 1, the second order (qaudratic) coefficient of the individualpixel fits is saved in an additional FITS file. All pixels that have second orderquadratic coefficient larger than the threshold parameter are flagged as non-linear and this resulting map of flags is written out as a third FITS file.

Products:

Name	Type	Description
IRD_ GAIN	FITS[Im(4)]	The linear coefficient of the Photon Transfer Curve (PTC) as image. The file contains the gain values in the first extensions. The second extension contains the bad pixels (static input bad pixels), the fourth extension contains the reduced chi-squared values. The third extension is not used and contains a zero image. The header contains the main gain measurement and its rms.
IRD_ NONLIN	FITS[Im(4)]	This product is only created if fitorder > 1. It is identical to the main product except that it contains the second (quadratic) coefficients of the pixel fits in the first extension.
IRD_ NONLIN_ BADPIX	FITS[Im(1)]	A simple image flagging all non linear pixels.

8.19 sph_ird_distortion_map

Purpose:

Creation of the total distortion map

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_DISTORTION_MAP_RAW	Raw data	No	1	500
IRD_MASTER_DARK	Calibration	Yes	0	1
IRD_FLAT_FIELD	Calibration	Yes	0	1
IRD_POINT_PATTERN	Calibration	Yes	0	1

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.distortion_map.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	distortion_map.fits	-
ird.distortion_map.point_table_filename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	distortion_point_table.fits	-
ird.distortion_map.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.distortion_map.clean_mean.reject_high	int	The clean mean reject pixels on high end.	0	0-20
ird.distortion_map.clean_mean.reject_low	int	The clean mean reject pixels on low end.	0	0-20
ird.distortion_map.threshold	double	The sigma above which point sources are detected.	3.0	0.0-200.0
ird.distortion_map.fitting_order	int	The polynomial fitting order for the 2D polynomial fit. This parameter is valid for both dimensions and it is not possible to have different fitting orders for the two dimensions.	3	2-8
ird.distortion_map.max_distortion	double	The maximal distortion allowed. Any points that are found to be further than this value from the expected value are excluded from the fitting procedure.	4.0	0.0-2000.0
ird.distortion_map.full-qc	bool	Full quality output wanted. Setting this to TRUE will create various QC images and also use the calculated distortion map to de-distort the input. When this flag is set, processing time of this recipe will increase measureably.	0	-
ird.distortion_map.user_cent	bool	the recipe finds the opitcal centre as the coordinates of the point that is closest to the geometrical centre of the point pattern.	0	-

Name	Type	Description	Default	Allowed vals.
ird.distortion_ map.cent_ left_ x	double	The opitcal centre of the left FOV. This is only used if the user_cent parameter is set to TRUE.	512.0	-
ird.distortion_ map.cent_ left_ y	double	The opitcal centre of the left FOV. This is only used if the user_cent parameter is set to TRUE.	512.0	-
ird.distortion_ map.cent_ right_ x	double	The opitcal centre of the right FOV. This is only used if the user_cent parameter is set to TRUE.	512.0	-
ird.distortion_ map.cent_ right_ y	double	The opitcal centre of the right FOV. This is only used if the user_cent parameter is set to TRUE.	512.0	-

Description:

This recipe creates a map of the distortion for the instrument. The raw framesare first reduced like standard science frames in field stabilised mode withoutdithering. The framecombination is simply done using a clean mean, mean or median combination. If given as input. a dark is subtracted and a flat field applied. The result frame is then analysed to detect point sources given the user detection threshold specified.Depending on whether a point pattern is given as one of the input frames or not,the recipe now either:

1. creates a new point pattern (if none was given) from the raw frames or
2. measures the distortion map comparing the observed point pattern with the inputpoint pattern provided.

In case that a new distortion map is created, this is done by

1. finding all points in the real image
2. making a guess of the opitcal axis. This is assumed to be the coordinates of thepoint closest to the geometrical centre of the point pattern.
3. shifting the input point pattern so that its most central point has the samecoordinates as the opitcal axis. This means that the central points on real and expected point pattern fallexactly ontop of each other.
4. determining the distance between each observed (detected) point and the closest point in the input table.
5. all points that have been found to be further than the max distortionvalue given as parameter to the recipe are removed.
6. The resulting distortion measurements are then used to calculate a polynomial fit to create a distortion map for all pixels.

The main product of the recipe is a mult-extension file that gives the distortion map for each IRDIS field of viewsperately. Other DRH recipes use the polynomial fit as stored in the header of extension 0 and 8 to apply the distortionmap.The recipe also produces a number of quality control files when requested to do so. The first is an image of the input point pattern, one total one and one each for the left and right FoVs. In addition the recipe uses the distortionmap that has been calculated in the main part of the recipe to correct the input processed raw image. This corrected input is written out as a full detector image as well as a left and right FoV subimages. To verify the distortion map is correct the recipe also produces residual distortion QC outputs when full QC output is requested. The absolute residual distortion images are named qc_residuals_left. fits and qc_residuals_right.fits. While these may show outliers, a high quality distortion measurement should yield residual images with typical values < 0.1.A stronger test of the quality of the distortion map quality can be made be feeding the full detector control image back intoa second run of the distortion map recipe. The resulting distortion map then gives the distortion residuals – and these shouldall be close to 0.

Products:

Name	Type	Description
IRD_ DISTORTION_ MAP	FITS[Im(16)]	The resulting distortion map. The distortion map is saved in a FITS file with a total of 16 extensions. The first 4 extensions contain values, badpixels, rms and weightmap for the distortion in the x direction and the next 4 extensions the same information for the distortion in the y direction. The first 8 extension contain the information for the left FOV the next 8 extension the information for the right FOV. Please also note that the image data is currently not used in subsecquent recipes – only polynomial fit parameters in the FITS header is used.
IRD_ POINT_ PATTERN	FITS[Table]	This frame is created only if no input point pattern was provided. The frame contains a new table giving the positions of all points found in the raw frames.

8.20 sph_ird_detector_persistence

Purpose:

Measure the detector persistence.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ DETECTOR_ PERSISTENCE_ OFF_ RAW	Raw data	No	2	Any
IRD_ DETECTOR_ PERSISTENCE_ ON_ SAT_ RAW	Raw data	No	1	Any
IRD_ DETECTOR_ PERSISTENCE_ ON_ UNSAT_ RAW	Raw data	No	1	Any
IRD_ MASTER_ DARK	Calibration	Yes	0	1
IRD_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
DATE	string	No	The creation date of the raw file.

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.detector_ persistence.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ird_ detector_ persistence_ map.fits	-
ird.detector_ persistence.fitorder	int	The order of the fit to use. Note that a fitorder > 2 can give unstable fitting results.	2	1-40
ird.detector_ persistence.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.detector_ persistence.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	0	0-20
ird.detector_ persistence.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	0	0-20
ird.detector_ persistence.threshold	double	The threshold for detection of illuminated regions. All regions with pixels above this value in the unsaturated image (with lamp on) are masked as illuminated regions in all other input frames.	10000.0	-

Description:

This recipe determines the detector persistence, by measuring the signal fall-offrate. The input raw frameset should contain frames taken with the illumination source on as well as off. Specifically, there should be at least one exposure containing a signigicant number of saturated pixels, at least one exposure containingilluminated (but not saturated pixels) and exposures with the source switched off. The exposures with illumination off should be taken in rapid succession immediately after the source is turned off.Frames are ordered in time sequence by the recipe, optionally a hotpixel mask from a master dark or a seperate image is used to mask bad pixels. As a first step, asimple thresholding algorithm is used on the illuminated but **unsaturated** imageto determine illuminated and unilluminated pixel sets, P_i and P_u. For each of the unilluminated frames,the mean for the unilluminated pixels $< P_u >$ issubtracted from the mean of the illuminated pixels giving $P(t) =< P_i > (t)- < P_u > (t)$.The series of $P(t)$ values is then fit assuming a polynomial behaviour in $1/t$, that is,assuming $P(t) = c_0 + c_1 \times 1/t + c_2 \times 1/t^2 + ...$ Up to which coefficient thefit is to be performed is set using the fitorder user parameter.A copy of the input illuminated but not saturated frame is saved as the main recipe product.The relevant persistence measurements are written as keywords into the product header.

Products:

Name	Type	Description
IRD_ DETECTOR_ PERSISTENCE	FITS[Imcube(4)]	A FITS cube with the fitting coefficients for each pixel. The fitting coefficients are for a polynomial fit of log(count) vs. log(time). Each plane in the image cube contains the values of one polynomial coefficient (starting with the constant term).

8.21 sph_ird_spectra_resolution

Purpose:

Check the spectra resolution

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ SPECTRA_ RESOLUTION_ RAW	Raw data	No	1	500
IRD_ FLAT_ FIELD	Calibration	Yes	0	1
IRD_ MASTER_ DARK	Calibration	Yes	0	1
IRD_ WAVECALIB	Calibration	No	1	1
IRD_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.

Name	Type	Description	Default	Allowed vals.
ird.spectra_ resolution.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	spectra_ resolution.fits	-
ird.spectra_ resolution.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.spectra_ resolution.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	0	0-20
ird.spectra_ resolution.clean_ mean.reject_ low	int	The clean mean reject pixels on lwo end.	0	0-20
ird.spectra_ resolution.threshold	double	Detectrion threshold for spectra lines	5.0	0.0-2000.0
ird.spectra_ resolution.wavelength_ line1	double	Wavelength of first line	987.72	-
ird.spectra_ resolution.wavelength_ line2	double	Wavelength of second line	1545.07	-
ird.spectra_ resolution.miny	int	The minimum y pixel corrdinates for acceptable pixels (should be set to 150 for grism mode).	0	0-1024
ird.spectra_ resolution.c1	double	The c1 coefficient for the wavelength fit. If no wavelength calibration file is provided the value given as parameter is used, otherwise it comes from the wavelength calibration file.	0.1	-
ird.spectra_ resolution.c2	double	The c2 coefficient for the wavelength fit.	-43.352	-
ird.spectra_ resolution.c3	double	The c3 coefficient for the wavelength fit.	149.723	-
ird.spectra_ resolution.c4	double	The c4 coefficient for the wavelenght fit.	82.442	-

Description:

This recipe monitors the spectra resolution for IRDIS spectroscopy mode. Theraw frames are reduced by dark subtraction, flat fielding and framecombination (assuming no dither). If provided, the pixel mask is used to mask out regionsthat are not to be included in the search for the line signal. The frame is then used to automatically detectthe spectra lines expected (two for each field of view). A thresholding (with the threshold as given bythe ird.spectra_resolution.threshold parameter) is performed. Since the imageis duplicated on the detector due to the double IRDIS optical path, a totalof 4 lines are expected to be found on the detector, but the recipe also works withless or with more lines. The resulting resolution power and other calculated quantitiesare then the average values for all detected lines.The central coordinates of the lines are written into the header of theproduct.

Products:

Name	Type	Description
IRD_ SPECTRA_ RESOLUTION	FITS[Im(4)]	The reduced spectra raw frames. The header contains keywords containing the information about the detected spectra lines. The FITS contains 4 extensions, with the image data, the badpixels, the rms and a weightmap.

8.22 sph_ird_ins_pol

Purpose:

Measure the instrument induced polarisation

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ INS_ POL_ RAW	Raw data	No	1	500
IRD_ MASTER_ DARK	Calibration	No	1	1
IRD_ FLAT_ FIELD	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS1 PAC X	double	No	The dithering position in X for the frame in pixels.
ESO INS1 PAC Y	double	No	The dithering position in Y for the frame in pixels.

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.ins_ pol.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ins_ pol.fits	-
ird.ins_ pol.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.ins_ pol.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	1	0-20
ird.ins_ pol.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	1	0-20

Description:

The recipe measures the polarisation introduced by the optical path between HWP2 and the detector. The raw framelist needs to contain frameswith two different settings for HWP2, so that there are frames with a switched polarisation. The frames with unswitched and the frames with switched-polarisations are sorted and then combined to give 2 resulting frames withthe switched(P_-) and unswitched(P_+) case. In both cases the dark and flat is applied. The product frame is then created by adding the frame P_-to the frame P_+. This then leaves the yields the instrument induced polarisation.

Products:

Name	Type	Description
IRD_ INS_ POL	FITS[Im(4)]	The polarisation frame. This frame shows the polarisation introduced by the instrument. The file has 4 extensions, the polarisation itself, the badpixel map, the rms error map and a weightmap.

8.23 sph_ird_ins_pol_eff

Purpose:

Measure the instrument polarisation efficiency

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ INS_ POL_ EFF_ RAW	Raw data	No	1	500
IRD_ MASTER_ DARK	Calibration	No	1	1
IRD_ FLAT_ FIELD	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS1 PAC X	double	No	The dithering position in X for the frame in pixels.
ESO INS1 PAC Y	double	No	The dithering position in Y for the frame in pixels.

Parameters:

Name	Type	Description	Default	Allowed vals.

Name	Type	Description	Default	Allowed vals.
ird.ins_pol_eff.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ins_pol_eff.fits	-
ird.ins_pol_eff.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.ins_pol_eff.clean_mean.reject_high	int	The clean mean reject pixels on high end.	1	0-20
ird.ins_pol_eff.clean_mean.reject_low	int	The clean mean reject pixels on low end.	1	0-20

Description:

This recipe measures the polarisation efficiency of the instrument. This is doneby measuring the polarisation of a 100% polarised source. The raw frames arereduced in the same way as for the usual polarisation science mode (see alsothe sph_ird_science_dpi recipe). The final frame is calculated as: P = (I_left - I_right) / (I_left + I_right).

Products:

Name	Type	Description
IRD_INS_POL_EFF	FITS[Im(4)]	The recuced polarisation image. It is saved as a FITS file with 4 extensions, the polarisation image, the badpixels, the rms error and a weightmap.

8.24 sph_ird_andromeda

Purpose:

Andromeda recipe.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ANDROMEDA_RAW	Raw data	No	1	Any
IRD_MASTER_DARK	Calibration	No	1	1
IRD_FLAT_FIELD	Calibration	No	1	1
IRD_DISTORTION_MAP	Calibration	Yes	0	1
IRD_STAR_CENTER	Calibration	Yes	0	1
IRD_FCTABLE	Calibration	Yes	0	Any
IRD_FILTER_TABLE	Calibration	Yes	0	1
IRD_PSF_REFERENCE	Calibration	No	1	1
IRD_STATIC_BADPIXELMAP	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS IRD DUAL FILTER LAMBDA LEFT	double	Yes	The central wavelength of the filter on left. Only needed if SDI requested.
ESO DRS IRD DUAL FILTER LAMBDA RIGHT	double	Yes	The central wavelength of the filter on right. Only needed if SDI requested.
ESO INS1 PAC X	double	No	The dithering position in X for the frame in pixels.
ESO INS1 PAC Y	double	No	The dithering position in Y for the frame in pixels.
ESO INS CPRT POSANG	double	Yes	The rotation angle of frame in degrees.

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.andromeda.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	andromeda.fits	-
ird.andromeda.left_ filename	string	The output filename for the left list after pre-processing. Only used if only_prep flag is set. Please also see the esorex documentatio for naming of output products.	left_ list.fits	-
ird.andromeda.right_ filename	string	The output filename for the left list after pre-processing. Only used if only_prep flag is set. Please also see the esorex documentatio for naming of output products.	right_ list.fits	-
ird.andromeda.keep_ fctable	bool	if set to TRUE the recipes internall created field center tables are not deleted.	0	-
ird.andromeda.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.andromeda.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	1	0-20
ird.andromeda.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	1	0-20
ird.andromeda.use_ sdi	bool	Flag to control usage of SDI	0	-
ird.andromeda.window_ minx	int	Window region andromeda is applied to.	428	0-1024
ird.andromeda.window_ miny	int	Window region andromeda is applied to.	428	0-1024
ird.andromeda.window_ maxx	int	Window region andromeda is applied to.	628	0-1024
ird.andromeda.window_ maxy	int	Window region andromeda is applied to.	628	0-1024
ird.andromeda.psf_ size	int	The size of the reference PSF. A central window of this size is extracted from the input PSF reference frame to create the PSF reference image to use by andromeda.	32	0-128

Name	Type	Description	Default	Allowed vals.
ird.andromeda.only_ prep	bool	Flag to switch off andromeda so only preperatory steps are performed: these are dark subtraction and flat fielding, frame cropping, frame centering and scaling (if SDI is on).	0	-
ird.andromeda.min_ ang_ sep	double	The minimum angle seperatrion to use to create the image pairs for image differencing.	1.0	0.0-45.0
ird.andromeda.rho_ min	double	The minimum radius to search for.	1.0	0.0-200.0
ird.andromeda.rho_ max	double	The maximum radius to search for.	10.0	0.0-200.0
ird.andromeda.filter_ radius	double	Filter radius for ADI framecombination. A non zero value leads to suppression of high frequencies in the fourier domain before framecombination. The value expresses the minimum unsuppressed frequency as fraction of total frequency domain radius (a value of 1 would suppress essentially all frequencies).	0.0	0.0-1.0

Description:

This recipe uses the Andromeda algorithm (Mugnier et al. 2008) for planet detection. The recipe has been implemented in C following the IDL script obtainedfrom L. Mugnier as much as possible. The basic reduction of raw frames follows that of the other IRDIS science recipes, in particulardark subtraction, flat fielding and frame centering is done as for the science_dbi recipe.Please see the science_dbi recipe for more details. Andromeda can also be used in combination with SDI by switching the use_sdi flag to TRUE.The current version is only a first attempt – please use with care.

Known Issues:

The recipe result if very sensitive to the input parameter choice and we believe this may indicate a bug somewhere. We also found that obtaining useful results on some input data is not possible.

Products:

Name	Type	Description
IRD_ ANDROMEDA	FITS[Im(4)]	The main science frame. The FITS file contains 4 extensions: the image, the badpixels, the rms error and a weightmap. All show the whole detector.

8.25 sph_ird_loci

Purpose:

LOCI recipe.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_LOCI_RAW	Raw data	No	1	Any
IRD_MASTER_DARK	Calibration	No	1	1
IRD_FLAT_FIELD	Calibration	No	1	1
IRD_DISTORTION_MAP	Calibration	Yes	0	1
IRD_STAR_CENTER	Calibration	Yes	0	1
IRD_FILTER_TABLE	Calibration	Yes	0	1
IRD_STATIC_BADPIXELMAP	Calibration	Yes	0	1
IRD_FCTABLE	Calibration	Yes	0	Any

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS IRD DUAL FILTER LAMBDA LEFT	double	Yes	The central wavelength of the filter on left. Only needed if SDI requested.
ESO DRS IRD DUAL FILTER LAMBDA RIGHT	double	Yes	The central wavelength of the filter on right. Only needed if SDI requested.
ESO INS1 PAC X	double	No	The dithering position in X for the frame in pixels.
ESO INS1 PAC Y	double	No	The dithering position in Y for the frame in pixels.
ESO INS CPRT POSANG	double	Yes	The rotation angle of frame in degrees.

Parameters:

Name	Type	Description	Default	Allowed vals.
ird.loci.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	loci.fits	-
ird.loci.left_filename	string	The output filename for the left list after pre-processing. Only used if only_prep flag is set. Please also see the esorex documentatio for naming of output products.	left_list.fits	-

Name	Type	Description	Default	Allowed vals.
ird.loci.right_ filename	string	The output filename for the left list after pre-processing. Only used if only_prep flag is set. Please also see the esorex documentatio for naming of output products.	right_ list.fits	-
ird.loci.keep_ fctable	bool	if set to TRUE the recipes internall created field center tables are not deleted.	0	-
ird.loci.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median.	1	0,1
ird.loci.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	1	0-20
ird.loci.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	1	0-20
ird.loci.use_ sdi	bool	Flag to control usage of SDI	0	-
ird.loci.na	double	The LOCI Na parameter	300.0	1.0-1000.0
ird.loci.ndelta	double	The LOCI Ndelta parameter	0.5	0.0-5.0
ird.loci.w	double	The LOCI w parameter (usually size of PSF)	2.0	0.0-50.0
ird.loci.g	double	The LOCI g parameter	1.0	0.0-5.0
ird.loci.minr	double	The minimum radius for the LOCI annulus	50.0	0.0-1000.0
ird.loci.maxr	double	The maximum radius for the LOCI annulus	200.0	0.0-1000.0
ird.loci.dr	double	The width of the segment annuli.	5.0	1.0-100.0
ird.loci.div_ scheme	int	The LOCI segment divisions scheme to use. 0 = NORMAL, 1 = FINE. .	1	0,1
ird.loci.filter_ radius	double	Filter radius for ADI framecombination. A non zero value leads to suppression of high frequencies in the fourier domain before framecombination. The value expresses the minimum unsuppressed frequency as fraction of total frequency domain radius (a value of 1 would suppress essentially all frequencies).	0.0	0.0-1.0

Description:

This is LOCI. LOCI is the >>locally optimized combination of images<< algorithm invented by Lafreniere and Marois.The SPHERE implementation follows the paper Lafraniere et al. (2007, ApJ, 660) very closely. Input parameters arenamed equivalently to the parameters as the appear in the paper.The preprocessing done before the actual LOCI algorithm is applied is the same as that for other IRDIS science recipes (e.g. science_dbi): the raw frames are dark subtracted, flat fielded and centered. It is also possible to runSDI before LOCI, by setting the use_sdi switch to TRUE. Please see the description for the science_dbi recipe for moredetails on the basic reductions.LOCI itself is implemented as in the original paper without an special tweaks. The step of subtracting the radial profile before LOCIis run as descibed in the paper is currently not implemented.The final output of the recipe is a LOCI image – since no special care is taken for normalisation etc. beware any flux determinations fromthis image.

Known Issues:

No support for radial profile subtraction.

Products:

Name	Type	Description
IRD_ LOCI	FITS[Im(4)]	The main science frame. The FITS file contains 4 extensions: the image, the badpixels, the rms error and a weightmap. All show the whole detector.

Chapter 9

IFS Pipeline Recipe Interfaces

9.1 sph_ifs_master_dark

Purpose:

Creation of the master dark frame

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_DARK_RAW	Raw data	No	1	Any

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.master_dark.outfilename	string	This parameter sets the filename that the product will be written out as. Please also see the esorex documentation about filename of products	master_dark.fits	-
ifs.master_dark.coll_alg	int	Set the collapse algorithm. The vaialable algorithms are: MEAN(0),MEDIAN (1),CLEAN_MEAN(2). Default is 2 for CLEAN_MEAN	2	0,1,2

Name	Type	Description	Default	Allowed vals.
ifs.master_ dark.badpixfilename	string	Controls the filename of the badpixel map.	static_ badpixels.fits	-
ifs.master_ dark.clean_ mean.reject_ high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low!	0	0-20
ifs.master_ dark.clean_ mean.reject_ low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low!	0	0-20
ifs.master_ dark.sigma_ clip	double	The sigma clipping value for static badpixel detection.Default is 5.0.	5.0	0.0-200.0
ifs.master_ dark.smoothing	double	The smoothing length (FWHM) to use for calculation of the large scale dark structures. Smoothing is needed for good hotpixel detection.	5.0	0.0-200.0
ifs.master_ dark.min_ acceptable	double	The minimum acceptable value. Any pixels with values below this are marked as bad.	-100.0	-
ifs.master_ dark.max_ acceptable	double	The maximum acceptable value. Any pixels with values above this are marked as bad.	1000.0	-
ifs.master_ dark.nskip	int	The number of planes in each input raw cube to skip. Removing the first planes in each dark cube in this way removes a spurious ramp effect at the beginning of each dark.	0	-

Description:

This recipe deals with the creation of the master dark calibration frame.Only raw frames are used in this recipe.The dark is created by combining the input raw frames using the collapsealgorithm specified (usually the clean_mean algorithm).After all input frames are combined in this way, the badpixels are determinedon the result. First a simple thresholding is applied using the parameters min_accepting and max_accepting.Now the resulting master dark is smoothed with a gaussian kernel of the FWHM specified in the smoothing user parameter, if this is set to a postive value. This smoothed version is subtractedfrom the master dark to remove large scale RMS variations.Then sigma clipping is used with the sigma user parameter. All pixels that are further than thespecified sigma value away from the mean are marked as bad.The resulting (unsmoothed) master dark frame is written out, including extensions for badpixels, RMS and an extension giving the number of input (raw) pixelsfor each output pixel.The hotpixel map is also written out as a separate parameter.

Products:

Name	Type	Description
IFS_ MASTER_ DARK	FITS[Im(4)]	The resulting master dark frame. This frame contains 4 different image extensions: the image, badpixels, the rms and the weightmap.

Name	Type	Description
IFS_ STATIC_ BADPIXELMAP	FITS[Im(1)]	An optionally written single extension image of the static badpixels. Note that the content is identical to the second extension in the master dark frame.

9.2 sph_ifs_master_detector_flat

Purpose:

Creation of the master detector flat frame

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ DETECTOR_ FLAT_ FIELD_ RAW	Raw data	No	2	500
IFS_ MASTER_ DARK	Calibration	Yes	0	1
IFS_ LARGE_ SCALE_ FLAT	Calibration	Yes	0	1
IFS_ PREAMP_ FLAT	Calibration	Yes	0	1
IFS_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.master_ detector_ flat.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	master_ detector_ flat.fits	-
ifs.master_ detector_ flat.save_ addprod	bool	Flag signalling hwether additional products should be saved, in this case a large scale flat, a preamp flat, and a hot pixels product.	0	-
ifs.master_ detector_ flat.lss_ outfilename	string	The output filename for the large scale flat product. Please also see the esorex documentation for naming of output products.	large_ scale_ flat.fits	-
ifs.master_ detector_ flat.preamp_ outfilename	string	The output filename for the preamplifier flat product. Please also see the esorex documentation for naming of output products.	preamp_ flat.fits	-

Name	Type	Description	Default	Allowed vals.
ifs.master_ detector_ flat.make_ badpix	bool	Controls if a seperate static badpixel map is requested for output.	0	-
ifs.master_ detector_ flat.badpixfilename	string	Controls the filename of the badpixel map, if requested for output. Ignored if no make_badpix is FALSE.@pd	dff_ badpixels.fits	-
ifs.master_ detector_ flat.robust_ fit	bool	Controls if fitting method is to be a robust linear fit. This will reduce the effect of cosmic rays and other temporary bad pixels. See e.g. Numerical Recipes for a description of the algorithm	0	-
ifs.master_ detector_ flat.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ifs.master_ detector_ flat.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	0	0-20
ifs.master_ detector_ flat.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	0	0-20
ifs.master_ detector_ flat.badpix_ lowtolerance	double	Threshold value for linearity badpixels. All pixels that have a flat field (slope) below this value will be flagged as bad.	0.1	-
ifs.master_ detector_ flat.badpix_ uptolerance	double	Threshold value for linearity badpixels. All pixels that have a flat field (slope) above this value will be flagged as bad.	10.0	-
ifs.master_ detector_ flat.badpix_ chisqtolerance	double	Threshold value for linearity badpixels. All pixels that have chi-squared value for the linear fit that is above this value will be flagged as bad	50.0	-
ifs.master_ detector_ flat.lambda	double	If this is set to a value > 0, the resulting master flat will be assigned the given calibration wavelength. In case that there are corresponding keywords present in the input raw frames, these are ignored in this case.	-1.0	-
ifs.master_ detector_ flat.smoothing_ length	double	The smooting length for the large scale flats.	10.0	-
ifs.master_ detector_ flat.smoothing_ method	int	The smooting method to use: 0 is square kernel using cpl_filter, 1 gauss kernel using FFT.	1	0,1

Description:

The detector flat field recipe for IFS is very similar to the instrument flatfield recipe for IRDIS. The recipe as described here uses input exposures taken with the narrow band or broad band calibration lamps.Several types of flat fields can be produced – in accordance with the calibration plan and the need to have seperate flat field components to provide maximal time stability andflat fielding accuracy. The recipe can be used to create a preamplifier correction flat (which can be used to remove the stripe structure caused by the pre amplifiers), a largescale flat field which is a smoothed flat field and hence only shows large scale structures,and a normal flat field. These master flats are used in all subsequent recipes that need to remove the pixel to pixel variation in the signal response of the detector. The recipe creates master calibration frames, using the input exposures which should be taken as described in the IFS calibration plan. The usual procedure to create a flat field is as follows. All raw frames are read in and dark subtracted. The frames are then corrected for the pre-amplifier variations. In case that a pre-amplifier correction flat was provided that frame is used,otherwise the correction is calculated directly from the raw frames (note that itis currently not possible to skip the pre amplifier correction altogether).This correction is a division operation rather than a subtraction.After this correction, the mean pixel value across the image is determined forall exposures. For every pixel $p = (x, y)$, a set of m_i,$v_i(x, y)$ data pairs are stored with m_i being the mean value of exposure i described above,and $v_i(x, y)$ being the pixel value for pixel $p(x, y)$ in exposure i.The flat field value is defined as the slope c_i of a linear fit F to the datam_i,v_i. The resulting slope represents the response of an indiviual pixel $p(x, y)$ to illumination relative to the detector mean response. The value will thus naturally be close to 1 and a division by that value will correct for a pixel's deviation from the average detector response.The fit itself is performed either using a maximum likelyhood methodor a robust fitting method which minimizes the sum of the absolute value of the deviations rather than the sum of the squares of the deviations (see e.g. Numerical Recipes for the algorithm). The robust fitting method will yield better results when significant outliers (e.g. due to cosmic rays) canbe expected, but does not allow anything but linear fits and can hence not be usedto assess detector non-linearity.In case that a large scale flat was provided as an input, this large scale flat is now divided out from the detector flat,removing the large scale dependence and leaving only the pixel-to-pixel variations. If no large scale was provided as input, no such correction is performed. The flat field values are saved as an image as the main product of the recipe.Aditionally, the recipe may also produce as output a map of all pixels that are identified as non-linear. The criteria for non-linearity are set by the userparameters and can be either pixels that have a flat field value outside specified bounds and/or pixels for which the linear fit produces a reduced chi-squared abovea given threshold value. For reliable non-linearity flagging using the reduced chi-squared fit many high quality input exposures are needed.In case that a non zero smoothing value was given, a large scale flat is also createdby smoothing the flat field with either a gaussian kernel using FFT or a square kernel usingthe specific CPL filter algorithm. Unless you know what you are doing leave the default methodhere which is the FFT smoothing.Choosing and creating the correct detector flat field for IFS is an important but tricky step. The best type of flat to use may depend on the exact science goals and final procedures and advicefor this step will be included in this manual at this point as soon as some on sky data for varioustype of science targets has been collected.Until that point the general advice is to:

1. calculate a preamp flat field using white lamp flat field data
2. calculate flat large scale fields for each of the coloured flat field lamps using the corresponding raw frames(but see below) and setting the lambda input parameter of this recipe in each case to the lamp wavelength. The preamp flat calculated at the previous step should be provided as input.
3. calculate a large scale flat for the white lamp (setting lambda=-1), again providing the preamp flat asadditional input
4. Use the preamp flat in addition to the all large scale flats when using recipes that need to correct the detector flat field (e.g. science recipes).

In some cases, it may be acceptable to use the white lamp raw frames as input to step 2 – this is effectively assumingthat there is no strong wavelength dependence of the detector flat field. Since taking flat field data is time consumingthis is a good option for all applications where an extremely accurate flat field correction is not required.

Products:

Name	Type	Description
IFS_ MASTER_ DETECTOR_ FLAT_ FIELD	FITS[Im(4)]	The flat field. This is saved as a FITS file with 4 extensions, the flat values, the badpixels (hotpixels and non-linear pixels), the rms error on the flat and a weightmap. Used if the lamp in use cannot be derived.
IFS_ MASTER_ DFF_ LONG1	FITS[Im(4)]	Same as above, produced from all input raw frames which had LAMP1 (1.020mum) switched on
IFS_ MASTER_ DFF_ LONG2	FITS[Im(4)]	Same as above, produced from all input raw frames which had LAMP2 (1.230mum) switched on
IFS_ MASTER_ DFF_ LONG3	FITS[Im(4)]	Same as above, produced from all input raw frames which had LAMP3 (1.300mum) switched on
IFS_ MASTER_ DFF_ LONG4	FITS[Im(4)]	Same as above, produced from all input raw frames which had LAMP4 (1.540mum) switched on
IFS_ MASTER_ DFF_ LONGBB	FITS[Im(4)]	Same as above, produced from all input raw frames which had either LAMP5 or LAMP6 (broad band) switched on.
IFS_ NON_ LINEAR_ PIXELMAP	FITS[Im(1)]	Optional output of all the non-linear pixels determined. All pixels as determined in this recipe using the ird.instrument_flat.badpix_*tolerance parameters.

9.3 sph_ifs_spectra_positions

Purpose:

Determinate of the spectra regions on detector

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ SPECPOS_ RAW	Raw data	No	1	Any
IFS_ INSTRUMENT_ FLAT_ FIELD	Calibration	Yes	0	1
IFS_ MASTER_ DARK	Calibration	Yes	0	1
IFS_ LENSLET_ MODEL	Calibration	Yes	0	1

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.spectra_ positions.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	spectra_ positions.fits	-
ifs.spectra_ positions.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean. A clean mean should be chosen to avoid contamination by cosmic rays.	2	0,1,2
ifs.spectra_ positions.clean_ mean.reject_ high	int	The clean mean reject pixels on high end. Choose a value above 0 to remove contamination by cosmics.	0	0-20
ifs.spectra_ positions.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	0	0-20
ifs.spectra_ positions.threshold	double	The threshold for detection of spectra regions (counts) If this is set to a negative value, the thresholding level is calculated automatically as the sum of the median value of the combined raw frames and the standard deviation on the combined raw frame.	-1.0	-
ifs.spectra_ positions.minpix	int	The minimum number of pixels a connected region has to contain to qualify as a spectra region.	25	-
ifs.spectra_ positions.angle	double	The rotation angle to assume for the lenslet array	-370.0	-
ifs.spectra_ positions.distortion	bool	Flag to set if distortion is to be measured. If set to true, the model is allowed to have distortion, otherwise it is rigid.	1	-
ifs.spectra_ positions.hmode	bool	Flag to set if default model should be Y-H (TRUE) instead of Y-J (FALSE). Note that this parameter is only effective if no input IFS lenslet model frame is given.	1	-

Description:

This recipe associates the IFS spectra with lenslets and associates pixels with wavelengths.The raw frames are reduced by dark subtracting and flat fielding and combined using the combination algorithm chosen (usually the clean mean algorithm). The flat field used can be eithera detector flat field, a flat field of the whole instrument (detector+IFU) or any other flatdeemed to be useful for this purpose. In most cases a detector flat field obtained with thebroad band calibration lamp seems the best choice.After a combined and reduced frame has been produced, a thresholding algorithm will determine the spectra regions. A second set of spectra regions is then determined using the IFS lenslet model.If a IFS lenslet model was provided in the form of a FITS file with the model parameters askeywords in its header, this model is used, otherwise the default model is used. The expectedspectra regions are compared with the actual one to determine a relative scale and offset. TheIFS lenslet model used for the expected pattern is subsequently updated to reflect the new, actualvalues of scale and position. This corrected model is then used to construct a pixel descriptiontable which is saved, together with the corrected lenslet model parameters, as the primaryproduct of this recipe.If desired, the model allows for a distortion. This may be crucial to ensure correct wavelengthcalibrations later in the calibration cascade. If distortion is allowed (using the user parameter)then the recipe will also fit a 2D polynomial of 4th order in x and y directions to the difference betweenmeasured and predicted spectra region centres. The coefficients of this polynomial model of thedistortion and incorporated in the product IFS lenslet model and will be used subsequently in allrecipes making use of the created lenslet model file.

Products:

Name	Type	Description
IFS_ SPECPOS	FITS[Im(6)]	The resulting pixel description table (PDT) written out as images. The PDT is written as a FITS file with 6 extensions, corresponding to: wavelength, spectra region id, lenslet id, wavelength width, second derivate of wavelength and illumination fraction. Currently the last two extensions are not used in any recipe.

9.4 sph_ ifs_ instrument_ flat

Purpose:

Determine the full instrument flat field OR the IFU flat

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ FLAT_ FIELD_ RAW	Raw data	No	1	Any
IFS_ WAVECALIB	Calibration	Yes	0	1
IFS_ SPECPOS	Calibration	Yes	0	1
IFS_ MASTER_ DARK	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ LONG1	Calibration	No	1	1
IFS_ MASTER_ DFF_ LONG2	Calibration	No	1	1
IFS_ MASTER_ DFF_ LONG3	Calibration	No	1	1
IFS_ MASTER_ DFF_ LONG4	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ LONGBB	Calibration	Yes	0	1
IFS_ PREAMP_ FLAT	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ SHORT	Calibration	Yes	0	1

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.instrument_ flat.iff_ filename	string	The output filename for the instrument flat product. Please also see the esorex documentation for naming of output products.	ifs_ instrument_ flat.fits	-
ifs.instrument_ flat.ifu_ filename	string	The output filename for the IFU flat product. Please also see the esorex documentation for naming of output products.	ifs_ ifu_ flat.fits	-
ifs.instrument_ flat.make_ badpix	bool	Controls if a seperate static badpixel map is requested for output.	0	-
ifs.instrument_ flat.nofit	bool	Allows polynomial fitting for flat field determination to be turned off. Instead the input raw frames will simply be collapsed with a median.	0	-
ifs.instrument_ flat.robust_ fit	bool	Controls if fitting method is to be a robust linear fit. This will reduce the effect of cosmic rays and other temporary bad pixels. See e.g. Numerical Recipes for a description of the algorithm	0	-
ifs.instrument_ flat.badpixfilename	string	Controls the filename of the badpixel map, if requested for output. Ignored if no make_badpix is FALSE.	iff_ badpixels.fits	-
ifs.instrument_ flat.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = clean mean, 3 = Weighted mean.	2	0,1,2
ifs.instrument_ flat.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	0	0-20
ifs.instrument_ flat.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	0	0-20
ifs.instrument_ flat.badpix_ lowtolerance	double	Threshold value for linearity badpixels. All pixels that have a flat field (slope) below this value will be flagged as bad	0.1	-
ifs.instrument_ flat.badpix_ uptolerance	double	Threshold value for linearity badpixels. All pixels that have a flat field (slope) above this value will be flagged as bad	10.0	-
ifs.instrument_ flat.badpix_ chisqtolerance	double	Threshold value for linearity badpixels. All pixels that have chi-squared value for the linear fit that is above this value will be flagged as bad	50.0	-

Name	Type	Description	Default	Allowed vals.
ifs.instrument_ flat.use_ illumination	bool	Controls if the illumination pattern of lenslets is to be taken into account in the cube creation or not. A low level wave-like structure can appear in the result if it is not applied. However, calculation of of the illumination fraction affects the performance of the recipe and so this option should only be enabled if the artefacts adversely affect the results. Note that there is a corresponding option on the ifs_science_dr recipe which should match the chosen option here.	0	-

Description:

This recipe creates the instrument flat for IFS.The recipe works in two modes: in the first mode, the raw frames from the calibration procedure are used to create a flat field on the detector which includes the effect of the detector response (the detector flat field is NOT divided out). The product created in this mode may be used as a flat field in the spectra_positions or wave_calib recipe.In the second mode, the pixel description table produced by the spectra_positions recipe and updated by wavelength calibration recipe is used as input together with the calibration raw frames to create a flat field of the IFU which has the effect of the detector response removed (i.e. it is divided by the detector flat).The mode to use is decided depending on the parameters set for this recipe andthe input files available.In the total flat field mode the recipe readsin the spectra positions file to set the illuminated regions. The recipe then constructs a flatfield in the same way as done for sph_ifs_master_detector_flat (subtracting the dark before). The resulting frames is then saved as the total instrument flat. Note that this framehas the same dimensions as the detector and is always dithering dependent.In the IFU flat field mode, the wavelength calibration file is used instead of the spectra positionstable. First the same steps are carried out as for the total flat mode. However, the wavlengthcalibration file is then used to first construct a wavelength dependent flat field (also sometimescalled a super flat field), making useof the series of master detector flats provided. This frame is then used to flat field the combined raw frames by dividing it out, to give a flat field containing only the IFU (lenslet) contribution. At this stage the frame is still for the detector itself. A lenslet description table is then constructed using the lenslet model as obtained from theheader of the input wavelength calibration frame. This lenslet description table nowcontains the extracted spectra data for all lenslets. These are then collapsed along thewavelength direction (taking the median values) to obtain a flat field value for alllenslets. The primary data product is written out as a viewable interpolated image(which is not generally used further in the cascade) and a table containing the flatfield values for all lenslets. This table is used in other recipes when the IFU flat fieldis to be applied.

Known Issues:

Products:

Name	Type	Description

Name	Type	Description
IFS_ INSTRUMENT_ FLAT_ FIELD	FITS[Im(4)]	The total instrument flat field. This is saved as a FITS file with 4 extensions, the flat values, the badpixels (hotpixels and non-linear pixels), the rms error on the flat and a weightmap.
IFS_ IFU_ FLAT_ FIELD	FITS[Im(4),Tab]	The IFU flat field. This is saved as a FITS file with 4 image extensions, the flat values, the badpixels (hotpixels and non-linear pixels), the rms error on the flat, a weightmap and 1 table extension containing the lenslet flat values.
IFS_ STATIC_ BADPIXELMAP	FITS[Im(1)]	Optional output of all the non-linear pixels determined. All pixels as determined in this recipe using the ird.instrument_flat.badpix_*tolerance parameters.

9.5 sph_ifs_wave_calib

Purpose:

Create the wavelength calibration data

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ WAVECALIB_ RAW	Raw data	No	1	Any
IFS_ SPECPOS	Calibration	No	1	1
IFS_ INSTRUMENT_ FLAT_ FIELD	Calibration	Yes	0	1
IFS_ MASTER_ DARK	Calibration	Yes	0	1

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.wave_ calib.outfilename	string	The output filename of the calibrated IFS model.	pdt_ wave_ calib.fits	-
ifs.wave_ calib.coll_ alg	int	The collapse algorithm to use to combine the input raw frames.	2	0,1,2

Name	Type	Description	Default	Allowed vals.
ifs.wave_ calib.number_ lines	int	The number of wavelength lines to fit.	3	2-5
ifs.wave_ calib.wavelength_ line1	double	The wavelength of the first line	0.98772	0.9-2.5
ifs.wave_ calib.wavelength_ line2	double	The wavelength of the second line	1.12371	0.9-2.5
ifs.wave_ calib.wavelength_ line3	double	The wavelength of the third line (only used if number_lines > 2)	1.30937	0.9-2.5
ifs.wave_ calib.wavelength_ line4	double	The wavelength of the fourth line (only used if number_lines > 3)	1.5451	0.9-2.5
ifs.wave_ calib.wavelength_ line5	double	The wavelength of the fourth line (only used if number_lines > 4)	1.5451	0.9-2.5
ifs.wave_ calib.line_ threshold	double	The threshold value to use for identifying spectral lines.	90.0	0.0-40000.0
ifs.wave_ calib.polyfit_ order	int	The order of the polynomial to use for the wavelength model. For example, if the order is 1, a linear model with constant dispersion is assumed.	2	-
ifs.wave_ calib.no_ spline_ interpol	bool	Do not use esoteric spline interpolation after wavelength fit. If true, the polynomial fit result will be directly inserted into the spectra without endpoint adaptation.	0	-
ifs.wave_ calib.fit_ window_ size	int	The tolerance around the predicted wavelength position to search for the actual maximum. This value should absolutely be smaller than the minimal distance between line wavelengths (in pixels).	4	1-10
ifs.wave_ calib.clean_ mean.reject_ high	int	Number of pixels to reject at high end for the clean mean combination method.	0	0-20
ifs.wave_ calib.clean_ mean.reject_ low	int	Number of pixels to reject at low end for the clean mean combination method.	0	0-20

Description:

This is the recipe responsible for calibrating the pixel to wavelength associations for the IFS. The approach taken for the IFS in SPHERE is model based: the initial model as created by the spectra positions calibration recipe is used as input and the observed wavelength calibration frames are used to modify this model, adjusting the pixel to wavelength associations. This approach assumes implicitly that there are not large discrepancies between the model and the actual wavelength associations.Before the wavelength associations are determined, the raw input frames are combined using the specified combination method (mean, clean mean or median). An optional dark is subtracted and the result divided by an optional flat field.The recipe then extracts a one dimensional spectrum for each spectral region (as found in the spectra positions recipe). Around each of the line l with wavelength w_l as specified in the user input, the flux weighted mean position is detetmined in a window around the expected position with a width as specified by the fit_window_size parameter:

$$y_l' = \sum_{y=y_{min}}^{y_{max}} s_y^2 \times y / \sum_{y=y_{min}}^{y_{max}} s_y^2, \tag{9.1}$$

where $y_{min} = y(w_l) - \Delta w$ if this is positive and $y_{min} = 0$ otherwise, and $y_{max} = y(w_l) + \Delta w$ if this is smaller than the total spectra length in pixels N and $y_{max} = N$ otherwise.The window region should be choses so as to avoid cases where there is none or more than one sharp line within the region of width $2\Delta w$ around the prediced pixel for the wavelength w_l.Once parameters y_l' have been determined for all $l = 1..n_l$ for a spectrum, a polxnomial fit is performed to the y_l' vs. w_l data. This polynomial is then used to fill in the wavelength associations for all pixels in the region of that spectrum.In cases when no fitting was possible (for example due to the fact that not enough identifiable lines were present in the spectra region) or if the resulting wavelength associations for the minimum and maximum wavelengths of the spectrum is different by more than 3 times the dispersion to the expected value, the assigned wavelengths of the model are used (even if a new associatin was found). In case no spectrum could be extraced in the first place, all wavelengths are set to zero and all pixels in the spectra region thereby marked as bad.When all spectra have been processed in this way, the final, now corrected, pixel description table is written out as the main recipe product. Several quality control keywords are provided in the header to help monitor the quality of the calibration.

Products:

Name	Type	Description
IFS_ WAVECALIB	FITS[Im(7)]	The calibrated pixel description table (PDT) written out as images. The PDT is written as a FITS file with 6 extensions, corresponding to: wavelength, spectra region id, lenslet id, wavelength width, second derivate of wavelength and illumination fraction. The last extension flags bad spectra for which no good fit was found. Currently the last three extensions are not used in any recipe.

9.6 sph_ifs_science_dr

Purpose:

Reduce science observations

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_SCIENCE_DR_RAW	Raw data	No	1	500
IFS_MASTER_DFF_LONG1	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG2	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG3	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG4	Calibration	Yes	0	1
IFS_MASTER_DFF_LONGBB	Calibration	Yes	0	1
IFS_PREAMP_FLAT	Calibration	Yes	0	1
IFS_MASTER_DFF_SHORT	Calibration	Yes	0	1
IFS_IFU_FLAT_FIELD	Calibration	Yes	0	1
IFS_MASTER_DARK	Calibration	Yes	0	1
IFS_STATIC_BADPIXELMAP	Calibration	Yes	0	1
IFS_WAVECALIB	Calibration	No	1	1
IFS_DISTORTION_MAP	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS2 DITH POSX	double	Yes	The dithering position in X for the frame in pixels.
ESO INS2 DITH POSY	double	Yes	The dithering position in Y for the frame in pixels.

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.science_dr.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ifs_science_dr.fits	-
ifs.science_dr.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 3 = Weighted mean.	3	0,1
ifs.science_dr.clean_mean.reject_high	int	The clean mean reject pixels on high end. Not currently used.	0	0-20
ifs.science_dr.clean_mean.reject_low	int	The clean mean reject pixels on low end. Not currently used.	0	0-20
ifs.science_dr.use_illumination	bool	Controls if the illumination pattern of lenslets is to be taken into account in the cube creation or not. A low level wave-like structure can appear in the result if it is not applied. However, calculation of of the illumination fraction affects the performance of the recipe and so this option should only be enabled if the artefacts adversely affect the results. Note that there is a corresponding option on the ifs_instrument_flat recipe which should match the chosen option here.	0	-

Name	Type	Description	Default	Allowed vals.
ifs.science_ dr.use_ adi	int	Use of ADI. If set to 0 angular differential imaging is not applied. If set to 1 then ADI is always applied if it is set to 2 then ADI is applied only of the total rotation in the input frames is larger than the angle given in the ifs.science_dr.min_adi_angle parameter.	2	0,1,2
ifs.science_ dr.min_ adi_ angle	double	Minimum angle for automatic ADI switch. When use_adi is set to automatic then the ADI is used iff the total rotation angle covered over the whole input is larger than the given value.	4.0	-
ifs.science_ dr.spec_ deconv	bool	If set to true, spectra deconvolution is used to combine the cubes.	1	-
ifs.science_ dr.outfilename	string	The basename for the output files (wihtout the .fits extension). Files will be named using a running number	ifs_ spec_ deconv	-
ifs.science_ dr.order	int	The order of the polynomial fit to be subtracted.	2	1-10
ifs.science_ dr.user_ cent	bool	If set to true, the user supplied center values are used, overriding the internally derived centers.	0	-
ifs.science_ dr.cx	double	If user_cent set to TRUE, this is the centre x coordinate to use. Coordinates are in FITS coords, so that the centre of a 291 times 291 pixel image is at 146.0,146.0	146.0	-
ifs.science_ dr.cy	double	If user_cent set to TRUE, this is the centre y coordinate to use. Coordinates are in FITS coords, so that the centre of a 291 times 291 pixel image is at 146.0,146.0	146.0	-
ifs.science_ dr.reflambda	double	The reference wavelength to use. Be careful with this parameter since the quality of the FFT scaling depends on this parameter. Scaling quality is generally better when choosing a value for the reference wavelength at the higher end of the specta range.	1.3	-
ifs.science_ dr.fwhm	double	A smoothing FWHM that will be used to improve the cosmetics. Smoothing is disabled if the parameter is 0 or negative.	-1.0	-

Description:

This is the science calibration recipe for IFS. The input raw observationframes are reduced by dark subtracting each one, if a master dark frame is given as input.Also, in case a prea-amplifier correction frame (basically for de-striping) the raw framesare corrected for the stripes.The large scale effects are then corrected by creating a so called super flat that combines the (large scale)flat fields taken with different colour lamps into a single flat field that takes the pixel towavelength correspondence into account using the information from the input wavelength calibrationfile.After dividing out the super flat, a broad band flat field that should have been created fromrecent data is divded out to remove the (small scale) flat variations that are very time dependent.Note that this broad band master flat field should already be corrected for the large scale flat structure.This means it must have been created using the master flat fieldrecipe with a large scale flat as input.The recipe allows automatic combination of images using the spec_deconv and simple_adi routines. These reduction steps can be selective switched on or off. Please see the documentation for the sph_ifs_sime_adi and sph_ifs_spec_deonv for details.To combine frames manually,the recipe sph_ifs_simple_adi or the sph_ifs_spec_deconv should be used.

Products:

Name	Type	Description
IFS_ SCIENCE_ DR	FITS[Imcube(4)]	The reduced science data as a wavelength cube.

9.7 sph_ifs_psf_reference

Purpose:

Image the PSF for future reference

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ PSF_ REFERENCE_ RAW	Raw data	No	1	500
IFS_ MASTER_ DFF_ LONG1	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ LONG2	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ LONG3	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ LONG4	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ LONGBB	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ SHORT	Calibration	Yes	0	1
IFS_ IFU_ FLAT_ FIELD	Calibration	No	1	1
IFS_ MASTER_ DARK	Calibration	No	1	1
IFS_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1
IFS_ WAVECALIB	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS2 DITH POSX	double	Yes	The dithering position in X for the frame in pixels.
ESO INS2 DITH POSY	double	Yes	The dithering position in Y for the frame in pixels.

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.psf_ reference.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ifs_ psf_ reference.fits	-
ifs.psf_ reference.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ifs.psf_ reference.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	0	0-20
ifs.psf_ reference.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	0	0-20
ifs.psf_ reference.use_ adi	bool	Controls the use of frame de-rotation. CURRENTLY NOT SUPPORTED!	0	-
ifs.psf_ reference.dither	bool	Controls the use of dithering.	0	-

Description:

This recipe is a specific purpose science reduction of a PSF reference image.The recipe is identical to the sph_ifs_science_dr recipe.

Products:

Name	Type	Description
IFS_ PSF_ REFERENCE	FITS[Imcube(4)]	The reduced science data as a wavelength cube.

9.8 sph_ifs_atmospheric

Purpose:

Measure the atmospheric effect

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ATMOSPHERIC_RAW	Raw data	No	1	500
IFS_MASTER_DFF_LONG1	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG2	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG3	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG4	Calibration	Yes	0	1
IFS_MASTER_DFF_LONGBB	Calibration	Yes	0	1
IFS_MASTER_DFF_SHORT	Calibration	Yes	0	1
IFS_IFU_FLAT_FIELD	Calibration	No	1	1
IFS_MASTER_DARK	Calibration	No	1	1
IFS_STATIC_BADPIXELMAP	Calibration	Yes	0	1
IFS_WAVECALIB	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS2 DITH POSX	double	No	The dithering position in X for the frame in pixels
ESO INS2 DITH POSY	double	No	The dithering position in Y for the frame in pixels

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.atmospheric.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ifs_atmospheric.fits	-
ifs.atmospheric.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ifs.atmospheric.clean_mean.reject_high	int	The clean mean reject pixels on high end.	0	0-20
ifs.atmospheric.clean_mean.reject_low	int	The clean mean reject pixels on low end.	0	0-20
ifs.atmospheric.use_adi	bool	Controls the use of frame de-rotation. CURRENTLY NOT SUPPORTED!	0	-
ifs.atmospheric.dither	bool	Controls the use of dithering.	0	-

Description:

This recipe masures the atmospheric effect. It is identical to the science recipe, except for tags and DFS related keywords.

Products:

Name	Type	Description
IFS_ATMOSPHERIC	FITS[Imcube(4)]	The reduced science data as a wavelength cube.

9.9 sph_ifs_astrometry

Purpose:

Measure plate scale and angle

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ASTROMETRY_RAW	Raw data	No	1	500
IFS_MASTER_DFF_LONG1	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG2	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG3	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG4	Calibration	Yes	0	1
IFS_MASTER_DFF_LONGBB	Calibration	Yes	0	1
IFS_MASTER_DFF_SHORT	Calibration	Yes	0	1
IFS_IFU_FLAT_FIELD	Calibration	No	1	1
IFS_MASTER_DARK	Calibration	No	1	1
IFS_STATIC_BADPIXELMAP	Calibration	Yes	0	1
IFS_WAVECALIB	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS2 DITH POSX	double	No	The dithering position in X for the frame in pixels
ESO INS2 DITH POSY	double	No	The dithering position in Y for the frame in pixels

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.astrometry.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ifs_astrometry.fits	-
ifs.astrometry.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ifs.astrometry.clean_mean.reject_high	int	The clean mean reject pixels on high end.	0	0-20
ifs.astrometry.clean_mean.reject_low	int	The clean mean reject pixels on low end.	0	0-20
ifs.astrometry.dither	bool	Flag to control dithering on or off.	0	-
ifs.astrometry.threshold	double	Threshold detection for companion (counts).	5.0	0.0-100.0
ird.astrometry.companion_x	double	The x position of companion relative to parent star in milliarcsec.	0.0	-400.0-400.0

Name	Type	Description	Default	Allowed vals.
ird.astrometry.companion_y	double	The y position of companion relative to parent star in milliarcsec.	0.0	-400.0-400.0

Description:

This recipe creates astrometry information. It does this by first reducingthe input raw frames, which should be of a binary star system, in the standard way. The recipe then attempts to detect both the star and thecompanion. From the relative position between star and companion therecipe determines the pixel scale and the rotation angle wrt. the northdirection. The input user parameter of star to companion relative position (in milliarcsec) as well as the instrument rotation angle keywords as set by the instrument control software (in the raw frames) are used forthis purpose. The data is written out as a collapsed science wavelength cube, with astromety information in the header.

Products:

Name	Type	Description
IFS_ ASTROMETRY	FITS[Im(4)]	The reduced science data, wavelength cube collapsed along wavelength direction.

9.10 sph_ifs_std_phot

Purpose:

Meausure the abolute photometric zero-point

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ STD_ PHOT_ RAW	Raw data	No	1	500
IFS_ MASTER_ DFF_ LONG1	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ LONG2	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ LONG3	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ LONG4	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ LONGBB	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ SHORT	Calibration	Yes	0	1
IFS_ IFU_ FLAT_ FIELD	Calibration	No	1	1
IFS_ MASTER_ DARK	Calibration	No	1	1
IFS_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1
IFS_ WAVECALIB	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS IFS CORONO	string	No	The keyword that specifies if the coronagraph is in or out.
ESO INS2 DITH POSX	double	No	The dithering position in X for the frame in pixels
ESO INS2 DITH POSY	double	No	The dithering position in Y for the frame in pixels

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.std_phot.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ifs_std_phot.fits	-
ifs.std_phot.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ifs.std_phot.clean_mean.reject_high	int	The clean mean reject pixels on high end.	0	0-20
ifs.std_phot.clean_mean.reject_low	int	The clean mean reject pixels on low end.	0	0-20
ifs.std_phot.dither	bool	Controls the use of dithering.	0	-
ifs.std_phot.threshold	double	The detection threshold for the standard star (sigma)	5.0	0.0-100.0
ifs.std_phot.max_rad	double	The maximum allowed distance from frame centre for the standard star (pixels)	300.0	0.0-500.0
ifs.std_phot.magnitude	double	The known magnitude of the photometric standard star.	3.0	-10.0-100.0

Description:

This recipe determines the photometric calibration. and measures the photometric zero point.The raw frames are first reduced as for the science_dr recipe. Only dithering of raw frames is supported here. The final reduced wavelength cube is collapsed along the wavelength axis. Theresulting frame is then analysed using a threshold algorithm to find the stellar component, allowingfor a small de-centering (using the max_rad parameter). When detected the count is measured andthe input known magnitude is used to calculate the photometric zero point as ZP=-2.5*log10(flux)-star_magnitude. The reduced science frame is written out along with akeyword for the measured total flux and the zero point.

Known Issues:

No sky background subtraction is currently done.

Products:

Name	Type	Description

Name	Type	Description
IFS_ STD_ PHOT	FITS[Im(4)]	The collapsed wavelength cube used for detection and measurement of the standard star flux. The result is saved as a FITS file with 4 extensions, image, baxpixels, rms and weightmap.

9.11 sph_ ifs_ flux_ calib

Purpose:

Determine the flux calibration of coronagraph

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ FLUX_ CALIB_ RAW	Raw data	No	1	500
IFS_ MASTER_ DFF_ LONG1	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ LONG2	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ LONG3	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ LONG4	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ LONGBB	Calibration	Yes	0	1
IFS_ MASTER_ DFF_ SHORT	Calibration	Yes	0	1
IFS_ IFU_ FLAT_ FIELD	Calibration	No	1	1
IFS_ MASTER_ DARK	Calibration	No	1	1
IFS_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1
IFS_ WAVECALIB	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS IFS CORONO	string	No	The keyword that specified if the coronagraph is in or out.
ESO INS2 DITH POSX	double	Yes	The dithering position in X for the frame in pixels
ESO INS2 DITH POSY	double	Yes	The dithering position in Y for the frame in pixels

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.flux_ calib.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ifs_ flux_ calib.fits	-

Name	Type	Description	Default	Allowed vals.
ifs.flux_ calib.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ifs.flux_ calib.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	0	0-20
ifs.flux_ calib.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	0	0-20
ifs.flux_ calib.dither	bool	Controls the use of dithering.	0	-
ifs.flux_ calib.threshold	double	The threshold (sigma) for detecting the stellar companion.	5.0	0.0-100.0
ifs.flux_ calib.min_ rad	double	Minimum search radius for detection of stellar companion (pixels)	5.0	0.0-500.0
ifs.flux_ calib.max_ rad	double	Maximum search radius for detection of stellar companion (pixels)	300.0	0.0-500.0

Description:

This recipe calibrates the effect of the coronagraph on the detected number ofcounts. For this purpose the raw frames with and without coronagraph are reducedsperately in the standard way (dark subtraction, flat fielding). After creation of thewavelength cubes, the wavelength axis is collapsed out, giving a single frame for each of the twocoronagraph settings. The final framesare then divided by each other (the frame without the coronagraph is divided bythe frame with the coronagraph) and the resulting product is saved along with thetotal fluxes measured in the different frames.

Products:

Name	Type	Description
IFS_ FLUX_ CALIB	FITS[Im(4)]	The reduced frame with the calibration keywords in header.

9.12 sph_ ifs_ sky_ flat

Purpose:

Create a sky flat

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_SKY_FLAT_RAW	Raw data	No	1	500
IFS_MASTER_DFF_LONG1	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG2	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG3	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG4	Calibration	Yes	0	1
IFS_MASTER_DFF_LONGBB	Calibration	Yes	0	1
IFS_MASTER_DFF_SHORT	Calibration	Yes	0	1
IFS_IFU_FLAT_FIELD	Calibration	No	1	1
IFS_MASTER_DARK	Calibration	No	1	1
IFS_STATIC_BADPIXELMAP	Calibration	Yes	0	1
IFS_WAVECALIB	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS2 DITH POSX	double	Yes	The dithering position in X for the frame in pixels.
ESO INS2 DITH POSY	double	Yes	The dithering position in Y for the frame in pixels.

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.sky_flat.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ifs_sky_flat.fits	-
ifs.sky_flat.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ifs.sky_flat.clean_mean.reject_high	int	The clean mean reject pixels on high end.	0	0-20
ifs.sky_flat.clean_mean.reject_low	int	The clean mean reject pixels on low end.	0	0-20
ifs.sky_flat.use_adi	bool	Controls the use of frame de-rotation. CURRENTLY NOT SUPPORTED!	0	-
ifs.sky_flat.dither	bool	Controls the use of dithering.	0	-

Description:

This recipe creates a sky flat. It is identical to sph_ifs_science_dr.

Known Issues:

Currently there is no proper determination of the flat field making use of the different illumination levels of the sky during dusk/dawn.

Products:

Name	Type	Description
IFS_SKY_FLAT	FITS[Imcube(4)]	The reduced science data as a wavelength cube.

9.13 sph_ifs_sky_cal

Purpose:

Measure the sky background

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_SKY_CAL_RAW	Raw data	No	1	500
IFS_MASTER_DFF_LONG1	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG2	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG3	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG4	Calibration	Yes	0	1
IFS_MASTER_DFF_LONGBB	Calibration	Yes	0	1
IFS_MASTER_DFF_SHORT	Calibration	Yes	0	1
IFS_IFU_FLAT_FIELD	Calibration	No	1	1
IFS_MASTER_DARK	Calibration	No	1	1
IFS_STATIC_BADPIXELMAP	Calibration	Yes	0	1
IFS_WAVECALIB	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS2 DITH POSX	double	Yes	The dithering position in X for the frame in pixels.
ESO INS2 DITH POSY	double	Yes	The dithering position in Y for the frame in pixels.

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.sky_cal.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ifs_sky_cal.fits	-
ifs.sky_cal.fit_filename	string	This parameter sets the filename that the product will be written out as. Please also see the esorex documentation about filename of products	ifs_sky_cal_smooth.fits	-
ifs.sky_cal.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2

Name	Type	Description	Default	Allowed vals.
ifs.sky_cal.clean_mean.reject_high	int	The clean mean reject pixels on high end.	0	0-20
ifs.sky_cal.clean_mean.reject_low	int	The clean mean reject pixels on low end.	0	0-20
ifs.sky_cal.use_adi	bool	Controls the use of frame de-rotation. CURRENTLY NOT SUPPORTED!	0	-
ifs.sky_cal.dither	bool	Controls the use of dithering.	0	-
ifs.sky_cal.fitorder	int	The fitting order to use for the 2D polynomial fitting.	2	1-7

Description:

This recipe measures the sky background. It is identical to sph_ifs_science_dr.

Products:

Name	Type	Description
IFS_SKY_CAL	FITS[Imcube(4)]	The reduced science data as a wavelength cube.

9.14 sph_ifs_gain

Purpose:

Measure the detector gain

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_GAIN_RAW	Raw data	No	4	Any

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.

Name	Type	Description	Default	Allowed vals.
ifs.gain.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ifs_ gain_ map.fits	-
ifs.gain.nonlin_ filename	string	The output filename for the nonlinearity map. Please also see the esorex documentation for naming of output products.	ifs_ nonlin_ map.fits	-
ifs.gain.nonlin_ bpixname	string	The output filename for the non linear bad pixel map. Please also see the esorex documentation for naming of output products.	ifs_ nonlin_ bpix.fits	-
ifs.gain.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ifs.gain.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	0	0-20
ifs.gain.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	0	0-20
ifs.gain.order	int	The fitting order to use, can be 1 (for linear only) or 2 (for lin+quadratic).	2	1-2
ifs.gain.lin_ tolerance	double	The allowed maximum absolute value of the second order of the polynomial fit. Any pixels that have an absolute value for the second order polynomial coefficient above this value are considered non-linear and marked as bad in the non-linearity map.	100.0	-
ifs.gain.preproc	bool	If set to TRUE, the raw frames are first processed to remove any offset trends within data cubes	1	-
ifs.gain.vacca	bool	Choose the special noise calculation by Vacca et al. (2004) that takes the number of fowler samples into account.	0	-

Description:

The gain recipe calculates the gain for the detector and derives a mask of nonlinear pixels. The input is assumed to be a series of data cubes, each containg a single extension with N > 3 planes that each contain a single exposure. The mean count for each input cube shoulld be differenteither by increasing the intensity of the illumination source or by using different exposuretimes. Note that in the latter case the recipe only produces the correct output if the detector gain is independent of the read out mode.The gain recipe, as well as the ron recipe, have a special optional preprocessing step, whichcorrects some possible bias due to readout electronics settings by first subtracting the median of each input cube from each image in the cube.The gain recipe offers to algorithms to calculate the gain, one straightforward fitting algorithm and a morecomplex fitting algorithm that takes the correct number of fowler samples into account. The second algorithmis switched on using the vacca user parameter and ispreferable for accurate gain determinations but can currently not be used to calcuate the detector non-linearity. It is therefore recommended to set the user parameter vacca to 1 whenan accurate gain measurement is needed but not non-linearity measurement is needed and 0 in all other cases.In particular for pure monitoring purposes to discover trends in the gain the simpler algorithm is sufficient. For both algorithms the general procedure is similar:The recipes calculates the gain by first collapsing allinput cubes to create a single mean image and variance image. The collapse algorithm specified(clean mean by default) and algorithm parameters are used for this process. Once a mean and varianceimage has been determined the median of the mean image and the corresponding variance is takenas one data point. The collection of input cubes then lead to a collection of data points of median andvariance, giving measurements of the variance vs. median relation for the detector. This is then fittedusing a polynomial of the specified order (usually 1 or 2). The slope of this curve is the inverseof the gain while the offset gives an estimate of the read out noise. Note that the read out noiseestimate obtained here may not be accurate. Please use the dedicated ron recipe to obtain a more accurate estimate of the RON. The estimates of gain and ronare written as keywords in the main recipe product FITS file.If the vacca parameter is set, the recipe corrects the fitting coefficients for the different noiseproperties expected for different fowler samples. For example, for double correlated reads this corrects the ron by a factor of 2.If the vacca parameter is not set. the recipe determines non linear pixels in a second step. This is done by performing the gain fitting procedure above for each individual pixel. The resulting map of the gain is the data in the first extension of the main productFITS file. Note that the pixel-by-pixel gain values are often very noisy and can not be used to obtainprecise gain measurements. Many exposures per input cube are needed to perform accurate pixel fitting.If the fitorder specified is larger than 1, the second order (qaudratic) coefficient of the individualpixel fits is saved in an additional FITS file. All pixels that have second orderquadratic coefficient larger than the threshold parameter are flagged as non-linear and this resulting map of flags is written out as a third FITS file.

Products:

Name	Type	Description
IFS_ GAIN	FITS[Im(4)]	The linear coefficient of the Photon Transfer Curve (PTC) as image. The file contains the gain values in the first extensions. The second extension contains the bad pixels (static input bad pixels), the fourth extension contains the reduced chi-squared values. The third extension is not used and contains a zero image. The header contains the main gain measurement and its rms.
IFS_ NONLIN	FITS[Im(4)]	This product is only created if fitorder >1. It is identical to the main product except that it contains the second (quadratic) coefficients of the pixel fits in the first extension.
IFS_ NONLIN_ BADPIX	FITS[Im(1)]	A simple image flagging all non linear pixels.

9.15 sph_ifs_detector_persistence

Purpose:

Measure the detector persistence.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ DETECTOR_ PERSISTENCE_ OFF_ RAW	Raw data	No	2	Any
IFS_ DETECTOR_ PERSISTENCE_ ON_ SAT_ RAW	Raw data	No	1	Any
IFS_ DETECTOR_ PERSISTENCE_ ON_ UNSAT_ RAW	Raw data	No	1	Any
IFS_ MASTER_ DARK	Calibration	Yes	0	1
IFS_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
DATE	string	No	The creation date of the raw file.

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.detector_ persistence.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ifs_ detector_ persistence_ map.fits	-
ifs.detector_ persistence.fitorder	int	The order of the fit to use. Note that a fitorder > 2 can give unstable fitting results.	2	1-40
ifs.detector_ persistence.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	1	0,1,2
ifs.detector_ persistence.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	0	0-20
ifs.detector_ persistence.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	0	0-20
ifs.detector_ persistence.threshold_ upper	double	The threshold for detection of illuminated regions. All regions with pixels above this value in the unsaturated image (with lamp on) are masked as illuminated regions in all other input frames.	40000.0	-
ifs.detector_ persistence.threshold_ lower	double	The threshold for detection of dark regions. All regions with pixels below this value in the unsaturated image (with lamp on) are masked as dark regions in all other input frames.	1000.0	-

Description:

This recipe determines the detector persistence, by measuring the signal fall-offrate. The input raw frameset should contain frames taken with the illumination source on as well as off. Specifically, there should be at least one exposure containing a signigicant number of saturated pixels, at least one exposure containingilluminated (but not saturated pixels) and exposures with the source switched off. The exposures with illumination off should be taken in rapid succession immediately after the source is turned off.Frames are ordered in time sequence by the recipe, optionally a hotpixel mask from a master dark or a seperate image is used to mask bad pixels. As a first step, asimple thresholding algorithm is used on the illuminated but **unsaturated** imageto determine illuminated and unilluminated pixel sets, P_i and P_u. For each of the unilluminated frames,the mean for the unilluminated pixels $< P_u >$ issubtracted from the mean of the illuminated pixels giving $P(t) =< P_i > (t)- < P_u > (t)$.The series of $P(t)$ values is then fit assuming a polynomial behaviour in $1/t$, that is,assuming $P(t) = c_0 + c_1 \times 1/t + c_2 \times 1/t^2 + ...$ Up to which coefficient thefit is to be performed is set using the fitorder user parameter.A copy of the input illuminated but not saturated frame is saved as the main recipe product.The relevant persistence measurements are written as keywords into the product header.

Products:

Name	Type	Description
IFS_ DETECTOR_ PERSISTENCE	FITS[Imcube(4)]	A FITS cube with the fitting coefficients for each pixel. The fitting coefficients are for a polynomial fit of log(count) vs. log(time). Each plane in the image cube contains the values of one polynomial coefficient (starting with the constant term).

9.16 sph_ifs_cal_background

Purpose:

Measure the instrument background.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ CAL_ BACKGROUND_ RAW	Raw data	No	1	Any
IFS_ MASTER_ DARK	Calibration	Yes	0	1

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.cal_ background.outfilename	string	This parameter sets the filename that the product will be written out as. Please also see the esorex documentation about filename of products	ifs_ cal_ background_ map.fits	-
ifs.cal_ background.fit_ filename	string	This parameter sets the filename that the product will be written out as. Please also see the esorex documentation about filename of products	ifs_ cal_ background_ map_ fit.fits	-
ifs.cal_ background.coll_ alg	int	Set the collapse algorithm. The vaialable algorithms are: MEAN(0),MEDIAN (1),CLEAN_MEAN(2). Default is 2 for CLEAN_MEAN.	2	0,1,2
ifs.cal_ background.clean_ mean.reject_ high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low!	1	0-20
ifs.cal_ background.clean_ mean.reject_ low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low!	1	0-20
ifs.cal_ background.sigma_ clip	double	The sigma clipping value for static badpixel detection.Default is 5.0.@	5.0	0.0-200.0
ifs.cal_ background.fit	bool	Flag to switch polynomial fitting on/off.	1	-
ifs.cal_ background.fitorder	int	The fitting order to use for the 2D polynomial fitting. Only used if the fit option is set to TRUE.	10	0-30
ifs.cal_ background.smooth	double	Smooting length to smooth the combined image. The value gives the FWHM of the gaussian that the combined image is convolved with to smooth it. Set the value to zero to switch smoothing off completely.	100.0	0.0-2000.0
ifs.cal_ background.nsamples	int	The number of samples to use for a polynomial 2D fit. Only used if the fit parameter is set.	1000	0-1000000

Description:

This recipe creates a summed, optionally dark subtracted frame that is mainly useful for background measurements.The raw input frames must carry the IFS_CAL_BACKGROUND_RAW tag. The dark frame is optional and will besubtracted from the resulting frame if porovided. Static badpixels are determined using the sigma_clipuser parmameter – an OR combination is used with the badpixels in the master dark frame if one if provided.The resulting background map is divided by the exposure time to give a result in counts per second. If a darkwas provided the exposure time of this dark is subtracted from the total exposure time before.In case the smooting parameter is set to a value above 0, the resuling combined frame is smoothed (after all badpixelshave been interpolated). If so desired, a 2D polynomial fit to the measured (possibly smoothed) bacground is also written outas a second product.

Products:

Name	Type	Description
IFS_CAL_BACKGROUND	FITS[Im(4)]	The total background. The frame is saved as a 4 extension FITS file with image, badpixels, rms and weightmap. The file gives the counts/seconds for each pixel.
IFS_CAL_BACKGROUND_FIT	FITS[Im(4)]	The total fitted background. The frame is saved as a 4 extension FITS file with image, badpixels, rms and weightmap. The units are counts/seconds for each pixel.

9.17 sph_ifs_distortion_map

Purpose:

Measure the lenslet array distortion

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_DISTORTION_MAP_RAW	Raw data	No	1	500
IFS_MASTER_DFF_LONG1	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG2	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG3	Calibration	Yes	0	1
IFS_MASTER_DFF_LONG4	Calibration	Yes	0	1
IFS_MASTER_DFF_LONGBB	Calibration	Yes	0	1
IFS_MASTER_DFF_SHORT	Calibration	Yes	0	1
IFS_IFU_FLAT_FIELD	Calibration	Yes	0	1
IFS_MASTER_DARK	Calibration	Yes	0	1
IFS_STATIC_BADPIXELMAP	Calibration	Yes	0	1
IFS_WAVECALIB	Calibration	No	1	1
IFS_POINT_PATTERN	Calibration	Yes	0	1
IFS_PREAMP_FLAT	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS2 DITH POSX	double	Yes	The dithering position in X for the frame in pixels
ESO INS2 DITH POSY	double	Yes	The dithering position in Y for the frame in pixels

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.distortion_ map.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ifs_ distortion_ map.fits	-
ifs.distortion_ map.ldt_ outfilename	string	The filename of the wavelength cube output. This is mainly for debugging purposes.	ifs_ distortion_ map_ ldt.fits	-
ifs.distortion_ map.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median.	1	0,1
ifs.distortion_ map.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	0	0-20
ifs.distortion_ map.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	0	0-20
ifs.distortion_ map.point_ table_ filename	string	The output filename for the point pattern table. Please also see the esorex documentation for naming of output products.	distortion_ point_ table.fits	-
ifs.distortion_ map.dither	bool	Controls the use of dithering.	0	-
ifs.distortion_ map.threshold	double	The threshold (sigma) for detecting the grind point sources.	3.0	0.0-200.0
ifs.distortion_ map.fitting_ order	int	The fitting order of the 2D polynomial fit to the distortions.	3	2-8
ifs.distortion_ map.max_ distortion	double	The maximum distortion to accept. Points above this value are removed before polynomial fitting.	4.0	0.0-2000.0
ifs.distortion_ map.self_ check	bool	Flag to set when a consistency self check is needed. An output is created containing the reduced input frames with the distortion map applied. Setting this option will double the execution time of this recipe!	0	-

Description:

This recipe measures the distortion of the lenslet grid. The inputraw frames are first reduced in the same way as for the science_drrecipe (allowing dither but no frame rotation), including dark subtractionand optional (if all flats provided) flat fielding. The resutling monochromaticimages are then collapsed along the wavelength direction giving a single image.This image is then analysed to detect the point sources using simplethresholding, taking the user parameter as the threshold value (sigmas).Now the way the recipe proceeds depends on the inputs. If no input point pattern has been provided, the recipe will construct one fromthe reduced images detected point sources. This will then be written outas a product. If a input point pattern was provided, the recipe will skip thisstep and proceed directly to the measurement of the distortion. Pleasenote that the distortion is measured in any case wich means that theoutput distortion should be zero in case that no input point pattern was provided.The distortion map is constructed by comparing the detected point sourcesposition with the expected point source position as provided in theinput point pattern. Each comparison is done with the closest point found andyields one distortion vector. Upon completion, all distortion vectorsthat have a length larger than the max_distortion specified as a userparameter will be removed before the x and y components of the distortionvectors will be fit with a 2D polynomial (of the user specified fitting order).To allow for easier quality control, the recipe provides a user flag for aself consistency check. If this flag is set, the recipe will now apply the measured distortion map to the input raw data and calculate any residual distortion left.This residual is written out as an extra FITS file – it should be visually inspectedto verify that the distortion map has been calculated with sufficient accuracy inthe detector region(s) of interest.

Products:

Name	Type	Description
IFS_ POINT_ PATTERN	FITS[Table]	The point pattern as obtained from the input images. Only written in case that an input point pattern was provided. This product may be used as reference input for future runs of this recipe.
IFS_ DISTORTION_ MAP	FITS[Im(8)]	The distortion map. The distortion map is saved as an 8 extension FITS file with the first 4 extensions containing the distortion in the x direction, the badpixels, the rms on the distortion in x and a weightmap. The second set of 4 extensions contain the same information but for the distortion in the y direction.

9.18 sph_ifs_dithering_effects

Purpose:

Measure the impact of dithering on wavelength calibration

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ DITHERING_ EFFECTS_ RAW	Raw data	No	1	Any
IFS_ SPECPOS	Calibration	No	1	1
IFS_ INSTRUMENT_ FLAT_ FIELD	Calibration	Yes	0	1
IFS_ MASTER_ DARK	Calibration	No	1	1
IFS_ WAVECALIB	Calibration	No	1	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS2 DITH POSX	double	No	The dithering position in X for the frame in pixels
ESO INS2 DITH POSY	double	No	The dithering position in Y for the frame in pixels

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.dithering_effects.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	pdt_dithering_effects.fits	-
ifs.dithering_effects.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ifs.dithering_effects.clean_mean.reject_high	int	The clean mean reject pixels on high end.	0	0-20
ifs.dithering_effects.clean_mean.reject_low	int	The clean mean reject pixels on low end.	0	0-20
ifs.dithering_effects.number_lines	int	The number of lines to use for wavelength calibration.	4	2-10
ifs.dithering_effects.wavelength_line1	double	The wavelength for line 1 (microns)	1.0	0.9-2.5
ifs.dithering_effects.wavelength_line2	double	The wavelength for line 2 (microns)	1.1	0.9-2.5
ifs.dithering_effects.wavelength_line3	double	The wavelength for line 3 (microns)	1.2	0.9-2.5
ifs.dithering_effects.wavelength_line4	double	The wavelength for line 4 (microns)	1.3	0.9-2.5
ifs.dithering_effects.waveength_line5	double	The wavelength for line 5 (microns)	1.4	0.9-2.5
ifs.dithering_effects.line_threshold	double	The threshold for line detectrion (sigma).	90.0	0.0-40000.0
ifs.dithering_effects.polyfit_order	int	The polynomial fitting order.	3	-

Description:

This recipe measure the effect of dithering on wavelength calibration. It does thisby reducing the raw frames as done for the wavelength calibration recipe. However,the raw frames are now at any arbitrary dithering position. The wavalengths are associated using a (software) shifted version of the input spectra positions file. This spectrapositions file is usually for the zeroth (reference) diterhing position – but thisis not a requirement. It can also be taken with the same dithering position asthe raw frames. Once a wavelength calibration has been performed from the rawinput frames the result is compared with a shifted version of the input wavelengthcalibration frame. The input wavelength calibration frame is shifted by the samealgorithm as used for all IFS science recipes to the same ditheringposition as the raw input frames. The difference between this and the measuredwavelengths is taken and saved as the output. This then gives an indication of howaccurate the software algorithm performs the wavelength calibration for dithered frames. Alternatively, this recipe can be used to monitor the dithering hardware.

Products:

Name	Type	Description
IFS_ DITHERING_ EFFECTS	FITS[Im(4)]	The effect of dithering on wavelength calibration. The FITS file shows in the first extension the difference between the measured wavelengths and those predicted by calculating the wavelengths from the zero position reference input frames to the dithering position of the raw (observed) frames. The second, third and fourth extensions contain badpixels, rms and weightmap.

9.19 sph_ifs_spec_deconv

Purpose:

The spectra deconvolution

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ SPEC_ DECONV_ RAW	Raw data	No	1	500

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.spec_ deconv.outfilename	string	The basename for the output files (wihtout the .fits extension). Files will be named using a running number	ifs_ spec_ deconv	-
ifs.spec_ deconv.order	int	The order of the polynomial fit to be subtracted.	2	1-10
ifs.spec_ deconv.user_ cent	bool	If set to true, the user supplied center values are used, overriding the internally derived centers.	0	-

Name	Type	Description	Default	Allowed vals.
ifs.spec_ deconv.cx	double	If user_cent set to TRUE, this is the centre x coordinate to use. Coordinates are in FITS coords, so that the centre of a 291 times 291 pixel image is at 146.0,146.0	146.0	-
ifs.spec_ deconv.cy	double	If user_cent set to TRUE, this is the centre y coordinate to use. Coordinates are in FITS coords, so that the centre of a 291 times 291 pixel image is at 146.0,146.0	146.0	-
ifs.spec_ deconv.reflambda	double	The reference wavelength to use. Be careful with this parameter since the quality of the FFT scaling depends on this parameter. Scaling quality is generally better when choosing a value for the reference wavelength at the higher end of the specta range.	1.3	-
ifs.spec_ deconv.fwhm	double	A smoothing FWHM that will be used to improve the cosmetics. Smoothing is disabled if the parameter is 0 or negative.	-1.0	-

Description:

This recipe implements a routine to perform a wavelength scaling and subtraction (similar to SDI) onwavelength cubes. Despite the recipe name, this is NOT a spectral deconvolution (the recipe name will be changedto reflect this before comissioning). Similar to the sph_simple_adi recipe, this recipe is not part of the basic cascade and only takes alreadyprocessed frames as input (as opposed to raw frames). The algorithm scales all images in a cube to a user specified reference wavelength. A polynomial fit is then performed along the wavelength direction for each spatial pixel. The order of the polynomial is a user parameter. This fit is then subtracted and the result images rescaled back to the original wavelenghts.This process effectively removes the speckle pattern. Processing is performed on a file by file base and there is a one to one correspondence between input and output frames.This recipe can therefore be run before a run of sph_simple_adi, allowing additional ADI reduction of framesthat have been processed by this recipe.

Products:

Name	Type	Description
IFS_ SPEC_ DECONV	FITS[Imcube(4)]	The reduced science data as a wavelength cube

9.20 sph_ifs_simple_adi

Purpose:

The simple ADI recipe

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_SIMPLE_ADI_RAW	Raw data	No	1	500

Raw frame keywords used:

none

Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.simple_adi.outfilename	string	The basename for the output files (wihtout the .fits extension). Files will be named using a running number	ifs_simple_adi	-
ifs.science_dr.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median.	2	0,1
ifs.simple_adi.transform_method	int	Transform method to use. 0 is FFT, 1 is CPL_WARP (interpolation).	0	0,1
ifs.simple_adi.filter_method	int	FFT filter method to use. 0 is none, 1 is top hat filter, 2 is Fermi fitler, 3 is Butterworth filter.	0	0,1,2,3
ifs.simple_adi.filter_rad	double	Radius for FFT top hat and Fermi filters. A non zero value leads to suppression of high frequencies in the fourier domain before framecombination. The value expresses the minimum unsuppressed frequency as fraction of total frequency domain radius (a value of 1 would suppress essentially all frequencies).	0.0	0.0-1.0
ifs.simple_adi.fermi_temp	double	The temperature parameter for the Fermi filter.	0.0	0.0-1.0
ifs.simple_adi.butter_pass	double	The pass band frequency for the Butterworth filter, as fraction of total frequency domain radius.	0.0	0.0-1.0
ifs.simple_adi.butter_stop	double	The stop band frequency for the Butterworth filter, as fraction of total frequency domain radius. This must be larger than the pass frequency.	0.0	0.0-1.0

Description:

This recipe performs a simple ADI routine on IFS spectroscopic data. The recipe is not part of the basic cascade and takes asinput only already processed frames, that have been created with the sph_ifs_science_dr recipe. The simple ADI process treats each wavelength section (plane) independently. For a specific wavelength, corresponding imagesare extracted from all the cubes and then combined using the ADI process and the result placed at the corresponding wavelengthin the result. The ADI process derotates the images by the angle as read from the header of the input file around the geometrical image center.The alogorithm to rotate the images by can be chosen using the user parameters, allowing also for some additional filter.After de-rotation the combination is performed either using a median or a mean.

Products:

Name	Type	Description
IFS_ SIMPLE_ ADI	FITS[Imcube(4)]	The reduced science data as a wavelength cube

Chapter 10

ZIMPOL Pipeline Recipe Interfaces

10.1 sph_zpl_preproc

Purpose:

Pre-processing of the zimpol raw data, polarimetric modes (utlity recipe).

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ PREPROC_ RAW	Raw data	No	1	Any

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DET READ CURNAME	string	Yes	KEYWORD NAME for the Detector Read Mode
ESO DET OUT1 X	int	Yes	KEWWORD NAME for X location of output, camera-1
ESO DET OUT1 Y	int	Yes	KEWWORD NAME for Y location of output, camera-1
ESO DET OUT1 NX	int	Yes	Output-1 data pixels in X, camera-1
ESO DET OUT1 NY	int	Yes	Output-1 data pixels in Y, camera-1
ESO DET OUT1 OVSCX	int	Yes	Output overscan pixels in X, camera-1
ESO DET OUT1 OVSCY	int	Yes	Output-1 overscan pixels in Y, camera-1
ESO DET OUT1 PRSCX	int	Yes	Output-1 prescan pixels in X, camera-1
ESO DET OUT1 PRSCY	int	Yes	Output-1prescan pixels in Y, camera-1
ESO DET OUT2 X	int	Yes	KEYWORD NAME for X location of output 2, camera-1
ESO DET OUT2 Y	int	Yes	KEYWORD NAME for Y location of output 2, camera-1
ESO DET OUT2 NX	int	Yes	Output-2 data pixels in X, camera-1
ESO DET OUT2 NY	int	Yes	Output 2 data pixels in Y, camera-1
ESO DET OUT2 OVSCX	int	Yes	Output 2 overscan pixels in X, camera-1
ESO DET OUT2 OVSCY	int	Yes	Output-2 overscan pixels in Y, camera-1
ESO DET OUT2 PRSCX	int	Yes	Output-2 prescan pixels in X, camera-1
ESO DET OUT2 PRSCY	int	Yes	Output-2 prescan pixels in Y, camera-1
ESO DET OUT1 X	int	Yes	KEWWORD NAME for X location of output, camera-2
ESO DET OUT1 Y	int	Yes	KEWWORD NAME for Y location of output, camera-2
ESO DET OUT1 NX	int	Yes	Output-1 data pixels in X, camera-2
ESO DET OUT1 OVSCX	int	Yes	Output-1 overscan pixels in X, camera-2
ESO DET OUT1 OVSCY	int	Yes	Output-1 prescan pixels in Y, camera-2
ESO DET OUT1 PRSCX	int	Yes	Output-1 prescan pixels in X, camera-2
ESO DET OUT1 PRSCY	int	Yes	Output-1 prescan pixels in Y, camera-2
ESO DET OUT2 X	int	Yes	KEWWORD NAME for X location of output, camera-2
ESO DET OUT2 Y	int	Yes	KEWWORD NAME for Y location of output, camera-2
ESO DET OUT2 NX	int	Yes	Output-2 data pixels in X, camera-2
ESO DET OUT2 NY	int	Yes	Output-2 data pixels in Y, camera-2
ESO DET OUT2 OVSCX	int	Yes	Output 2 overscan pixels in X, camera-2
ESO DET OUT2 OVSCY	int	Yes	Output-2 overscan pixels in Y, camera-2
ESO DET OUT2 PRSCX	int	Yes	Output-2 prescan pixels in X, camera-2
ESO DET OUT2 PRSCY	int	Yes	Output-2 prescan pixels in Y, camera-2

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.preproc.outfilename_cam1	string	The output postfix-filename of the pre-processed raw data for CAMERA-1.	preproc_cam_1.fits	-
zpl.preproc.outfilename_cam2	string	The output postfix-filename of the pre-processed raw data for CAMERA-2.	preproc_cam_2.fits	-

Description:

This recipe performs pre-processing steps for the raw data in the polarimetric modes. The pre-processing is an utility recipe and should be only used by off-line data reduction! The raw frame in the polarimetric modes are two extensions fits-file format:

- first extension represents data cube of NDITS frames from camera-1 for a given DIT, including prescan /overscan area of 2 ADUs;
- second extension represents data cube of NDITS zimpol frames from camera-2 for a given DIT, including prescan / overscan area of 2 ADUs.

No other frame is used in this recipe.In all polarimetric ZIMPOL modes (P1,P2,P3) detector mode is always double-phase mode. In the double-phase detector mode one single ZIMPOL-exposure is output of two consecutive images/frames from one CCD:

- the 1 image is the k-th ZIMPOL frame recorded at phase one=Phase 0
- the 2 image is the k+1 ZIMPOL frame recorded at phase one=Phase PI

Each frame contains 2-interlaced sub-frames, storing 2 complimentary polarization component images. The input raw cube frame should carry SPH_ZPL_TAG_PREPROC_RAW tag is read first and then the following pre-processing steps are performed:

1. extract each camera from the each extension (»two camera cubes«);
2. combine the 2 detector segments (ADU) into a single image »trim away« prescan/overscan areas from images;
3. compute the mean overscan bias level from the overscan areas;
4. cut junk rows for Phase 0 (one bottom and one upper »binned pixel« row);
5. cut junk rows for Phase PI (two upper »binned pixel«);
6. split into even and odd sub-frames;
7. for each two single raw images (phase 0 and pi) create a plane with 4 extensions;
8. create an output fits files with four images extension and one binary table extension with the computedmean values of the overscan bias level and its rms (8 cols).

Since the zimpol frame is square, splitting the two sub-frames yields to the 1:2 aspect ratio. The output product is two fits-cube files (camera-1 and camera-2) of the ZPL EXP format specified as follows:

- phase zero odd sub-frame image;
- phase zero even sub-frame image;
- phase PI odd sub-frame image;
- phase PI even sub-frame;
- table of mean overscan bias level values and its rms (8 cols):
  - **[1-col, 2-col]** ADU1 ZERO PHASE OVSC MEAN & ADU1 ZERO PHASE OVSC RMS
  - **[3-col, 4-col]** ADU2 ZERO PHASE OVSC MEAN & ADU2 ZERO PHASE OVSC RMS
  - **[5-col, 6-col]** ADU1 PI PHASE OVSC MEAN & ADU1 PI PHASE OVSC RMS
  - **[7-col, 8-col]** ADU2 PI PHASE OVSC MEAN & ADU2 PI PHASE OVSC RMS

These pre-processing products may be used in all subsequent polarimetric recipes.

Products:

Name	Type	Description
ZPL_ PREPROC	FITS[Im(4),Bt(1)]	The output product is two fits-cube files with 5 extensions (for camera-1 and camera-2) of the ZPL EXP format specified as follows: - phase zero odd sub-frame image; - phase zero even sub-frame image; - phase PI odd sub-frame image; - phase PI even sub-frame; table of mean overscan bias level values and its rms (8 cols): - ADU1 ZERO PHASE OVSC MEAN, ADU1 ZERO PHASE OVSC RMS, ADU2 ZERO PHASE OVSC MEAN, ADU2 ZERO PHASE OVSC RMS, - ADU1 PI PHASE OVSC MEAN, ADU1 PI PHASE OVSC RMS, ADU2 PI PHASE OVSC MEAN, ADU2 PI PHASE OVSC RMS

10.2 sph_zpl_preproc_imaging

Purpose:

Pre-processing of the zimpol raw data, imaging mode (utility recipe).

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ PREPROC_ IMAGING_ RAW	Raw data	No	1	Any

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DET READ CURNAME	string	Yes	KEYWORD NAME for the Detector Read Mode
ESO DET OUT1 X	int	Yes	KEWWORD NAME for X location of output, camera-1
ESO DET OUT1 Y	int	Yes	KEWWORD NAME for Y location of output, camera-1
ESO DET OUT1 NX	int	Yes	Output-1 data pixels in X, camera-1
ESO DET OUT1 NY	int	Yes	Output-1 data pixels in Y, camera-1
ESO DET OUT1 OVSCX	int	Yes	Output overscan pixels in X, camera-1
ESO DET OUT1 OVSCY	int	Yes	Output-1 overscan pixels in Y, camera-1
ESO DET OUT1 PRSCX	int	Yes	Output-1 prescan pixels in X, camera-1
ESO DET OUT1 PRSCY	int	Yes	Output-1prescan pixels in Y, camera-1
ESO DET OUT2 X	int	Yes	KEYWORD NAME for X location of output 2, camera-1
ESO DET OUT2 Y	int	Yes	KEYWORD NAME for Y location of output 2, camera-1
ESO DET OUT2 NX	int	Yes	Output-2 data pixels in X, camera-1
ESO DET OUT2 NY	int	Yes	Output 2 data pixels in Y, camera-1
ESO DET OUT2 OVSCX	int	Yes	Output 2 overscan pixels in X, camera-1
ESO DET OUT2 OVSCY	int	Yes	Output-2 overscan pixels in Y, camera-1
ESO DET OUT2 PRSCX	int	Yes	Output-2 prescan pixels in X, camera-1
ESO DET OUT2 PRSCY	int	Yes	Output-2 prescan pixels in Y, camera-1
ESO DET OUT1 X	int	Yes	KEWWORD NAME for X location of output, camera-2
ESO DET OUT1 Y	int	Yes	KEWWORD NAME for Y location of output, camera-2
ESO DET OUT1 NX	int	Yes	Output-1 data pixels in X, camera-2
ESO DET OUT1 OVSCX	int	Yes	Output-1 overscan pixels in X, camera-2
ESO DET OUT1 OVSCY	int	Yes	Output-1 prescan pixels in Y, camera-2
ESO DET OUT1 PRSCX	int	Yes	Output-1 prescan pixels in X, camera-2
ESO DET OUT1 PRSCY	int	Yes	Output-1 prescan pixels in Y, camera-2
ESO DET OUT2 X	int	Yes	KEWWORD NAME for X location of output, camera-2
ESO DET OUT2 Y	int	Yes	KEWWORD NAME for Y location of output, camera-2
ESO DET OUT2 NX	int	Yes	Output-2 data pixels in X, camera-2
ESO DET OUT2 NY	int	Yes	Output-2 data pixels in Y, camera-2
ESO DET OUT2 OVSCX	int	Yes	Output 2 overscan pixels in X, camera-2
ESO DET OUT2 OVSCY	int	Yes	Output-2 overscan pixels in Y, camera-2
ESO DET OUT2 PRSCX	int	Yes	Output-2 prescan pixels in X, camera-2
ESO DET OUT2 PRSCY	int	Yes	Output-2 prescan pixels in Y, camera-2

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.preproc_ imaging.outfilename_ cam1	string	The output postfix-filename of the pre-processed raw data for CAMERA-1.	preproc_ imaging_ cam_ 1.fits	-
zpl.preproc_ imaging.outfilename_ cam2	string	The output postfix-filename of the pre-processed raw data for CAMERA-2.	preproc_ imaging_ cam_ 2.fits	-

Description:

This recipe performs pre-processing steps for the raw data in the imaging modes.The pre-processing is an utility recipe and should be only used by off-line data reduction. The raw frame cube in the imaging mode is two extensions fits-files with the following format:

- first extension represents data cube of NDITS frames from camera-1 for a given DIT, including overscan area of 2 ADUs;
- second extention represents data cube of NDITS zimpol frames from camera-2 for a given DIT, including overscan area of 2 ADUs.

No other frame is used in this recipe.In the imaging detector mode each raw frame contains 2-interlaced sub-frames, but the useful imaging component is only kept in the first sub-frame because the second one is masked (it can be considered as a dark current).The input raw cube frame, which should carry SPH_ZPL_TAG_PREPROC_IMAGING_RAW tag, are read first and then the following pre-processing steps are performed:

1. extract each camera from the each extension (»two camera cubes«);
2. combine the 2 detector segments (ADU) into a single image »trim away« prescan/overscan areas;
3. split into even and odd sub-frames;
4. for each initial raw image create a plane with two extensions (informative component - intensity, dark current);
5. create an output fits files with two images extension and one binary table extension with the computedmean values of the overscan bias level and its rms (4 cols).

Since the zimpol frame is square, splitting the two sub-frames yields to the 1:2 aspect ratio. The pre-processing imaging output product is written out in the two fits-cube files (camera-1 and camera-2) withthe ZPL EXP IMAGING format specified as follows:

- odd sub-frame image (informative component);
- even sub-frame image (dark current component);
- table of mean overscan bias level values and its rms (4 cols):
  - **[1-col, 2-col]** ADU1 OVSC MEAN & ADU1 OVSC RMS
  - **[3-col, 4-col]** ADU2 OVSC MEAN & ADU2 OVSC RMS

These pre-processing imaging products may be used in all subsequent imaging recipes.

Products:

Name	Type	Description
ZPL_ PREPROC_ IMAGING	FITS[Im(2),Bt(1)]	The output product is two fits-cube files with 3 extensions (for camera-1 and camera-2) of the ZPL EXP IMAGING format specified as follows: - odd sub-frame image; - sub-frame image; table of mean overscan bias level values and its rms (4 cols): - ADU1 OVSC MEAN, ADU1 OVSC RMS, ADU2 OVSC MEAN, ADU2 OVSC RMS

10.3 sph_zpl_master_bias

Purpose:

Create master bias, polarization modes.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_BIAS_RAW	Raw data	Yes	0	Any
ZPL_BIAS_PREPROC	Calibration	Yes	0	Any
ZPL_BIAS_PREPROC_CAM1	Calibration	Yes	0	Any
ZPL_BIAS_PREPROC_CAM2	Calibration	Yes	0	Any

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if the pre-processed data are used. As the format of the zimpol pre-processed data is complicated, this keyword was introduced in order to garantee that the pre-processed input frames are polarimetric pre-processed data, produced by the sph_zpl_preproc utility recipe. The value of this keyword is set up to >>SPH PC PREPROC ZPL EXP<<. Note: if raw data are used (default), then all keywords needed for the pre-processing recipe (see sph_zpl_preproc) must be presented in the raw data.

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.master_bias.outfilename	string	The output filename for the product of the camera-1/2. Please also see the esorex documentation for naming of output products.	zpl_master_bias.fits	-
zpl.master_bias.outfilename_cam1	string	The output filename for the product of the camera-1. Please also see the esorex documentation for naming of output products.	zpl_master_bias_cam1.fits	-
zpl.master_bias.outfilename_cam2	string	The output filename for the product of the camera-2. Please also see the esorex documentation for naming of output products.	zpl_master_bias_cam2.fits	-

Name	Type	Description	Default	Allowed vals.
zpl.master_bias.subtract_overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.master_bias.coll_alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median, 2 = Clean Mean. Default is 2 = Clean Mean	2	0,1,2
zpl.master_bias.coll_alg.clean_mean.reject_high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.master_bias.coll_alg.clean_mean.reject_low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.master_bias.clean_mean.sigma	double	The number of pixels to reject when combining frames in sigma from median. NOT SUPPORTED YET!	5.0	0.0-200.0
zpl.master_bias.sigma_clip	double	The sigma clipping value for static badpixel detection. Default is 0 (no sigma clipping).	0.0	0.0-200.0
zpl.master_bias.keep_intermediate	bool	Flag to set if intermediate date must be saved, namely pre-processed and overscan pre-processed subtracted data (FALSE)	0	-
zpl.preproc.outfilename_cam1	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-1.	preproc_cam1.fits	-
zpl.preproc.outfilename_cam2	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-2.	preproc_cam2.fits	-

Description:

This recipe creates a master bias calibration product for all polarization modes.The input frames might be either bias raw frames with the ZPL_BIAS_RAW tag or pre-processed bias frames, which should carrythe ZPL_BIAS_PREPROC_CAM1 and/or ZPL_BIAS_PREPROC_CAM2 tags. No other frames are used in this recipe. If input frames are raw frames then the master bias recipe first performs the pre-processing step for all input frames (raw cubes),creating corresponding pre-processed frames (cubes) for both ZIMPOL cameras (see also sph_zpl_preproc for the detailed description of the pre-processing). The master bias for each camera is then created by combining pre-processed frames (= all planes) from the pre-processed cube(s) using a specified collapse algorithm (usually the clean_mean algorithm, defined as a default one).If the flag >>subtract_overscan<< is not set up to 0, then the recipe subtracts (before combining) the overscan bias levelof the pre-processed cube(s) individually for each plane. Otherwise, the overscan subtraction step is skipped. (The calulcated overscan bias levels – >>ADU1 mean overscan value<< from the left area of the image, and >>ADU2 mean overscan value<< from the right area of the image – for each phase (0 and pi) are saved anyway as a binary table in the pre-processed cube(s)).After all pre-processed frames (all 4 zpl exposure sub-frames) are combined in this way,the badpixel maps are determined on the result, using a simple sigma clipping algorithm.It sets the bad/hot pixels to be all those that are further than the >>specified sigma x the total RMS<< of the whole image away from the image median. The resulting master dark frames for both cameras are written out in the SPH QUAD (16 extensions) format specified as follows:

1. zpl exp phase zero odd sub-frame:
   - master bias image,
   - badpixel-map,
   - ncomb-map,
   - rms-map;
2. zpl exp phase zero even sub-frame:
   - master bias image
   - badpixel-map
   - ncomb-map
   - rms-map;
3. zpl exp phase PI odd sub-frame:
   - master bias image,
   - badpixel-map,
   - ncomb-map,
   - rms-map;
4. zpl exp phase PI even sub-frame:
   - master bias image,
   - badpixel-map,
   - ncomb-map,
   - rms-map.

Note that the default parameter for the sigma clipping >>sigma_clip<< is set up to 0. In this case the recipe willnot detect >>hot/bad pixels<<, so all pixels will be considered as good ones in the product. Usually, the zpl exposure frames have the two vertical pixel stripes with strong >>bias<< signal. If the >>sigma_clip<< parameter is not 0, these pixels will be detected as bad ones and will be excluded from the subsequent treatment in the sphere pipeline (according to the sphere pipeline concept the detected bad pixel is marked as a bad in the badpixel-map and its rms value set to the 1e10 in the rms-map). Therefore, using the master bias for all subsequent polarimetric recipes in the defalut case (no sigma clipping), will preserve the signal in the vertical pixels stripes. The master polarimetric bias products may be used in the all subsequent polarimetric recipes.

Products:

Name	Type	Description

Name	Type	Description
ZPL_ MASTER_ BIAS	FITS[Im(16)]	The resulting master bias frame is of the QUAD IMAGE format. This frame contains 16 image extensions (4 master frames), grouped by the following order: zpl exp phase zero odd sub-frame master bias image, badpixel-map, ncomb-map and rms-map; zpl exp phase zero even sub-frame master bias image, badpixel-map, ncomb-map and rms-map; zpl exp phase PI odd sub-frame master bias image, badpixel-map, ncomb-map and rms-map; zpl exp phase PI even sub-frame master bias image, badpixel-map, ncomb-map and rms-map.
ZPL_ MASTER_ BIAS_ CAM1	FITS[Im(16)]	The resulting master bias frame is of the QUAD IMAGE format. This frame contains 16 image extensions (4 master frames), grouped by the following order: zpl exp phase zero odd sub-frame master bias image, badpixel-map, ncomb-map and rms-map; zpl exp phase zero even sub-frame master bias image, badpixel-map, ncomb-map and rms-map; zpl exp phase PI odd sub-frame master bias image, badpixel-map, ncomb-map and rms-map; zpl exp phase PI even sub-frame master bias image, badpixel-map, ncomb-map and rms-map.
ZPL_ MASTER_ BIAS_ CAM2	FITS[Im(16)]	The resulting master bias frame is of the QUAD IMAGE format. This frame contains 16 image extensions (4 master frames), grouped by the following order: zpl exp phase zero odd sub-frame master bias image, badpixel-map, ncomb-map and rms-map; zpl exp phase zero even sub-frame master bias image, badpixel-map, ncomb-map and rms-map; zpl exp phase PI odd sub-frame master bias image, badpixel-map, ncomb-map and rms-map; zpl exp phase PI even sub-frame master bias image, badpixel-map, ncomb-map and rms-map.

10.4 sph_zpl_master_bias_imaging

Purpose:

Create master bias, imaging mode.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_BIAS_IMAGING_RAW	Raw data	Yes	0	Any
ZPL_BIAS_IMAGING_PREPROC	Calibration	Yes	0	Any
ZPL_BIAS_IMAGING_PREPROC_CAM1	Calibration	Yes	0	Any
ZPL_BIAS_IMAGING_PREPROC_CAM2	Calibration	Yes	0	Any

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if the pre-processed data are used. As the format of the zimpol data is complicated, this keyword is introduced in order to garantee that the input frames are imaging pre-processed data, produced by the sph_zpl_preproc recipe which added this keyword automatically. The value of this keyword is set up to >>SPH PC PREPROC ZPL EXP IMAGING<<. Note: if raw data are used (default), then all keywords neede for the pre-processing recipe (see sph_zpl_preproc_imaging) must be presented in the raw data.

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.master_bias_imaging.outfilename	string	The output filename for the product for the camera-1/2. Please also see the esorex documentation for naming of output products.	zpl_master_bias_imaging.fits	-
zpl.master_bias_imaging.outfilename_cam1	string	The output filename for the product for the camera-2. Please also see the esorex documentation for naming of output products.	zpl_master_bias_imaging_cam1.fits	-
zpl.master_bias_imaging.outfilename_cam2	string	The output filename for the product for the camera-2. Please also see the esorex documentation for naming of output products.	zpl_master_bias_imaging_cam2.fits	-

Name	Type	Description	Default	Allowed vals.
zpl.master_bias_imaging.subtract_overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.master_bias_imaging.coll_alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median, 2 = Clean Mean. Default is 2 = Clean Mean	2	0,1,2
zpl.master_bias_imaging.coll_alg.clean_mean.reject_high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.master_bias_imaging.coll_alg.clean_mean.reject_low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.master_bias_imaging.clean_mean.sigma	double	The number of pixels to reject when combining frames in sigma from median. NOT SUPPORTED YET!	5.0	0.0-200.0
zpl.master_bias_imaging.sigma_clip	double	The sigma clipping value for static badpixel detection. Default is 0 (no sigma clipping).	0.0	0.0-200.0
zpl.master_bias_imaging.keep_intermediate	bool	Flag to set if intermediate date must be saved, namely pre-processed and overscan pre-processed subtracted data	0	-
zpl.preproc.outfilename_cam1	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-1.	preproc_cam1.fits	-
zpl.preproc.outfilename_cam2	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-2.	preproc_cam2.fits	-

Description:

This recipe creates the master bias calibration frame for the imaging mode.The input frames might be either bias raw frames with the ZPL_BIAS_IMAGING_RAW tag or pre-processed bias frames, which should carry the ZPL_BIAS_IMAGING_PREPROC_CAM1 and/or ZPL_BIAS_IMAGING_PREPROC_CAM2 tags. No other frames are used in this recipe.If input frames are raw frames then the master bias recipe first performs the pre-processing step for all input frames (raw cubes),creating corresponding pre-processed frames (cubes) for both ZIMPOL cameras (see also sph_zpl_preproc_imaging for the detailed description of the pre-processing). The master bias for each camera is then created by combining pre-processed frames (= all planes) from the imaging pre-processed cube(s) using a specified collapse algorithm (usually the clean_mean algorithm, defined as a default one).If the flag >>subtract_overscan<< is not set up to 0, the recipe subtracts (before combining) the overscan bias level from the pre-pocessed cube(s) individually for each plane.Otherwise, the overscan subtraction step is skipped.(The overscan bias level – >>ADU1 mean overscan value<< from the left area of the image and >>ADU2 mean overscan value<< from the right area of the image – for odd and even sub-frames are saved anyway as a binary table in the imaging pre-processed cube(s)). After all pre-processed frames (all 4 zimpol exposure sub-frames) are combined in this way,the badpixel maps are determined on the result, using a simple sigma clipping algorithm.It sets the bad/hot pixels to be all those that are further than the >>specified sigma x the total RMS<< of the whole image away from the image median. The resulting master dark frames for both cameras are written out in the DOUBLE IMAGE (8 extensions) format specified as follows:

1. odd sub-frame (informative component):
   - master bias image,
   - badpixel-map,
   - ncomb-map,
   - rms-map;
2. even sub-frame (dark current component):
   - master bias image,
   - badpixel-map,
   - ncomb-map,
   - rms-map.

Note that the default parameter for the sigma clipping >>sigma_clip<< is set up to 0. In this case the recipe willnot detect >>hot/bad pixels<<, so all pixels will be considered as good ones in the product. Usually, the zpl exposure imaging frames have the two vertical pixel stripes with strong >>bias<< signal. If the >>sigma_clip<< parameter is not 0, these pixels will be detected as bad ones and will be excluded from the subsequent treatment in the sphere pipeline (according to the sphere pipeline concept the detected bad pixel is marked as a bad in the badpixel-map andits rms value set to the 1e10 in the rms-map). Therefore, using the master bias for all subsequent imaging recipes in the defalut case (no sigma clipping), will preserve the signal in the vertical pixels stripes. The master imaging bias products are used in all subsequent imaging recipes.

Products:

Name	Type	Description
ZPL_MASTER_BIAS_IMAGING	FITS[Im(8)]	The resulting master bias frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions (2 master frames), grouped by the following order: odd sub-frame master bias image (informative), badpixel-map, ncomb-map and rms-map; even sub-frame master bias image (dark current), badpixel-map, ncomb-map and rms-map;
ZPL_MASTER_BIAS_IMAGING_CAM1	FITS[Im(8)]	The resulting master bias frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions (2 master frames), grouped by the following order: odd sub-frame master bias image (informative), badpixel-map, ncomb-map and rms-map; even sub-frame master bias image (dark current), badpixel-map, ncomb-map and rms-map;

Name	Type	Description
ZPL_ MASTER_ BIAS_ IMAGING_ CAM2	FITS[Im(8)]	The resulting master bias frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions (2 master frames), grouped by the following order: odd sub-frame master bias image (informative), badpixel-map, ncomb-map and rms-map; even sub-frame master bias image (dark current), badpixel-map, ncomb-map and rms-map;

10.5 sph_zpl_master_dark

Purpose:

Create master dark, polarization modes.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ DARK_ RAW	Raw data	Yes	0	Any
ZPL_ DARK_ PREPROC	Calibration	Yes	0	Any
ZPL_ DARK_ PREPROC_ CAM1	Calibration	Yes	0	Any
ZPL_ DARK_ PREPROC_ CAM2	Calibration	Yes	0	Any
ZPL_ MASTER_ BIAS	Calibration	Yes	0	1
ZPL_ MASTER_ BIAS_ CAM1	Calibration	Yes	0	1
ZPL_ MASTER_ BIAS_ CAM2	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if the pre-processed data are used. As the format of the zimpol pre-processed data is complicated, this keyword was introduced in order to garantee that the pre-processed input frames are polarimetric pre-processed data, produced by the sph_zpl_preproc utility recipe. The value of this keyword is set up to >>SPH PC PREPROC ZPL EXP<<. Note: if raw data are used (default), then all keywords needed for the pre-processing recipe (see sph_zpl_preproc) must be presented in the raw data.

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.master_dark.outfilename	string	The output filename for the product of the camera-1/2. Please also see the esorex documentation for naming of output products.	zpl_master_dark.fits	-
zpl.master_dark.outfilename_cam1	string	The output filename for the product of the camera-1. Please also see the esorex documentation for naming of output products.	zpl_master_dark_cam1.fits	-
zpl.master_dark.outfilename_cam2	string	The output filename for the product of the camera-2. Please also see the esorex documentation for naming of output products.	zpl_master_dark_cam2.fits	-
zpl.master_dark.subtract_overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.master_dark.coll_alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median, 2 = Clean Mean. Default is 2 = Clean Mean	2	0,1,2
zpl.master_dark.coll_alg.clean_mean.reject_high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.master_dark.coll_alg.clean_mean.reject_low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.master_dark.clean_mean.sigma	double	The number of pixels to reject when combining frames in sigma from median. NOT SUPPORTED YET!	5.0	0.0-200.0
zpl.master_dark.sigma_clip	double	The sigma clipping value for static badpixel detection. Default is 0 (=inf).	0.0	0.0-200.0
zpl.master_dark.keep_intermediate	bool	Flag to set if intermediate date must be saved, namely pre-processed and overscan pre-processed subtracted data (FALSE)	0	-
zpl.preproc.outfilename_cam1	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-1.	preproc_cam1.fits	-
zpl.preproc.outfilename_cam2	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-2.	preproc_cam2.fits	-

Description:

This recipe creates the master dark calibration frame for the polarization mode.The input frames might be either dark raw frames with the ZPL_DARK_RAW tag or pre-processed dark frames, which should carrythe ZPL_DARK_PREPROC_CAM1 and/or ZPL_DARK_PREPROC_CAM2 tags, and master bias frames (if any) with the

ZPL_MASTER_BIAS_CAM1 and/or ZPL_MASTER_BIAS_CAM2 tags.If input frames are raw frames then the master dark recipe first performs the pre-processing step for all input frames (raw cubes),creating corresponding pre-processed frames (cubes) for both ZIMPOL cameras (see also sph_zpl_preproc for the detailed description of the pre-processing step). The master dark for each camera is then created by combining pre-processed frames (= all planes) from the pre-processed cube(s) using a specified collapse algorithm (usually the clean_mean algorithm, defined as a default one).If the flag >>subtract_overscan<< is not set up to 0, then the recipe subtracts (before combining) the overscan bias levelof the pre-processed cube(s) individually for each plane. Otherwise, the overscan subtraction step is skipped. (The calulcated overscan bias levels – >>ADU1 mean overscan value<< from the left area of the image, and >>ADU2 mean overscan value<< from the right area of the image – for each phase (0 and pi) are saved anyway as a binary table in the pre-processed cube(s)).After all pre-processed frames are combined in this way (all 4 zimpol exposure sub-frames),the badpixel maps are determined on the results, using a simple sigma clipping algorithm.It sets the bad/hot pixels to be all those that are further than the >>specified sigma x the total RMS<< of the whole image away from the image median. The resulting master dark frames are subtracted by master bias frames and the products of both camerasare written out in the QUAD IMAGE(16 extensions) format specified as follows:

1. zpl exp phase zero odd sub-frame:
    - master dark image,
    - badpixel-map,
    - ncomb-map,
    - rms-map;
2. zpl exp phase zero even sub-frame:
    - master dark image
    - badpixel-map
    - ncomb-map
    - rms-map;
3. zpl exp phase PI odd sub-frame:
    - master dark image,
    - badpixel-map,
    - ncomb-map,
    - rms-map;
4. zpl exp phase PI even sub-frame:
    - master dark image,
    - badpixel-map,
    - ncomb-map,
    - rms-map.

This master polarimetric dark products can be used in the all subsequent polarimetric recipes.

Products:

Name	Type	Description
ZPL_ MASTER_ DARK	FITS[Im(16)]	The resulting master dark frame is of the QUAD IMAGE format. This frame contains 16 image extensions (4 master frames), grouped by the following order: zpl exp phase zero odd sub-frame master dark image, badpixel-map, ncomb-map and rms-map; zpl exp phase zero even sub-frame master dark image, badpixel-map, ncomb-map and rms-map; zpl exp phase PI odd sub-frame master dark image, badpixel-map, ncomb-map and rms-map; zpl exp phase PI even sub-frame master dark image, badpixel-map, ncomb-map and rms-map.
ZPL_ MASTER_ DARK_ CAM1	FITS[Im(16)]	The resulting master dark frame is of the QUAD IMAGE format. This frame contains 16 image extensions (4 master frames), grouped by the following order: zpl exp phase zero odd sub-frame master dark image, badpixel-map, ncomb-map and rms-map; zpl exp phase zero even sub-frame master dark image, badpixel-map, ncomb-map and rms-map; zpl exp phase PI odd sub-frame master dark image, badpixel-map, ncomb-map and rms-map; zpl exp phase PI even sub-frame master dark image, badpixel-map, ncomb-map and rms-map.
ZPL_ MASTER_ DARK_ CAM2	FITS[Im(16)]	The resulting master dark frame is of the QUAD IMAGE format. This frame contains 16 image extensions (4 master frames), grouped by the following order: zpl exp phase zero odd sub-frame master dark image, badpixel-map, ncomb-map and rms-map; zpl exp phase zero even sub-frame master dark image, badpixel-map, ncomb-map and rms-map; zpl exp phase PI odd sub-frame master dark image, badpixel-map, ncomb-map and rms-map; zpl exp phase PI even sub-frame master dark image, badpixel-map, ncomb-map and rms-map.

10.6 sph_zpl_master_dark_imaging

Purpose:

Create master dark, imaging mode.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ DARK_ IMAGING_ RAW	Raw data	Yes	0	Any
ZPL_ DARK_ IMAGING_ PREPROC	Calibration	Yes	0	Any
ZPL_ DARK_ IMAGING_ PREPROC_ CAM1	Calibration	Yes	0	Any
ZPL_ DARK_ IMAGING_ PREPROC_ CAM2	Calibration	Yes	0	Any
ZPL_ MASTER_ BIAS_ IMAGING	Calibration	Yes	0	1
ZPL_ MASTER_ BIAS_ IMAGING_ CAM1	Calibration	Yes	0	1
ZPL_ MASTER_ BIAS_ IMAGING_ CAM2	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if pre-processed data are used. As the format of the zimpol data is complicated, this keyword is introduced in order to garantee that the input frames are imaging pre-processed data, produced by the sph_zpl_preproc recipe which added this keyword automatically. The value of this keyword is set up to >>SPH PC PREPROC ZPL EXP IMAGING<<. Note: if raw data are used (default), then all keywords needed for the pre-processing recipe (see sph_zpl_preproc_imaging) must be presented in the raw data.

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.master_ dark_ imaging.outfilename	string	The output filename for the product for the camera-1/2. Please also see the esorex documentation for naming of output products.	zpl_ master_ dark_ imaging.fits	-
zpl.master_ dark_ imaging.outfilename_ cam1	string	The output filename for the product for the camera-1. Please also see the esorex documentation for naming of output products.	zpl_ master_ dark_ imaging_ cam1.fits	-

Name	Type	Description	Default	Allowed vals.
zpl.master_ dark_ imaging.outfilename_ cam2	string	The output filename for the product for the camera-2. Please also see the esorex documentation for naming of output products.	zpl_ master_ dark_ imaging_ cam2.fits	-
zpl.master_ dark_ imaging.subtract_ overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.master_ dark_ imaging.coll_ alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median, 2 = Clean Mean. Default is 2 = Clean Mean	2	0,1,2
zpl.master_ dark_ imaging.coll_ alg.clean_ mean.reject_ high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.master_ dark_ imaging.coll_ alg.clean_ mean.reject_ low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.master_ dark_ imaging.clean_ mean.sigma	double	The number of pixels to reject when combining frames in sigma from median. NOT SUPPORTED YET!	5.0	0.0-200.0
zpl.master_ dark_ imaging.sigma_ clip	double	The sigma clipping value for static badpixel detection. Default is 0 (=inf).	0.0	0.0-200.0
zpl.master_ dark_ imaging.keep_ intermediate	bool	Flag to set if intermediate date must be saved, namely pre-processed and overscan pre-processed subtracted data	0	-
zpl.preproc.outfilename_ cam1	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-1.	preproc_ cam1.fits	-
zpl.preproc.outfilename_ cam2	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-2.	preproc_ cam2.fits	-

Description:

This recipe creates master dark calibration frames for the imaging modes.The input frames might be either dark raw frames with the ZPL_DARK_IMAGING_RAW tag or pre-processed dark frames, which should carry the ZPL_DARK_IMAGING_PREPROC_CAM1 and/or ZPL_DARK_IMAGING_PREPROC_CAM2 tags, and master bias frames (if any) with the ZPL_MASTER_BIAS_IMAGING_CAM1 and/or ZPL_MASTER_BIAS_IMAGING_CAM2 tags.If input frames are raw frames then the master dark recipe first performs the pre-processing step for all input frames (raw cubes),creating corresponding pre-processed frames (cubes) for both ZIMPOL cameras (see also sph_zpl_preproc_imaging for the detailed description of the pre-processing step). The master imaging dark for each camera is then created by combining pre-processed frames (= all planes) from imaging pre-processed cube(s) using a specified collapse algorithm (usually the clean_mean algorithm, defined as a default one).If the flag >>subtract_overscan<< is not set up to 0, the recipe subtracts (before combining) the overscan bias level from the pre-pocessed cube(s) individually for each plane. Otherwise, the overscan subtraction step is skipped.(The overscan bias level – >>ADU1 mean overscan value<< from the left area of the image and >>ADU2 mean overscan value<< from the right area of the image – for odd and even sub-frames are saved anyway as a binary table in the imaging pre-processed cube(s)).After all pre-processed frames (all 2 zimpol imaging exposure sub-frames) are combined in this way,the badpixel maps are determined on the results, using a simple sigma clipping algorithm.It sets the bad/hot pixels to be all those that are further than the >>specified sigma x the total RMS<< of the whole image away from the image median. The resulting master dark imaging frames for both cameras are subtractedby the corresponding master bias imaging calibrations and written out in the DOUBLE IMAGE (8 extensions) format specified as follows:

1. odd sub-frame (informative component):
   - master dark image,
   - badpixel-map,
   - ncomb-map,
   - rms-map;
2. even sub-frame (dark current component):
   - master dark image,
   - badpixel-map,
   - ncomb-map,
   - rms-map.

The master imaging dark products are used in the all subsequent imaging recipes.

Products:

Name	Type	Description
ZPL_ MASTER_ DARK_ IMAGING	FITS[Im(8)]	The resulting master dark frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions (2 master frames), grouped by the following order: - odd sub-frame master dark image (informative), badpixel-map, ncomb-map and rms-map; - even sub-frame master dark image (dark current), badpixel-map, rms-map and rms-map.
ZPL_ MASTER_ DARK_ IMAGING_ CAM1	FITS[Im(8)]	The resulting master dark frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions (2 master frames), grouped by the following order: - odd sub-frame master dark image (informative), badpixel-map, ncomb-map and rms-map; - even sub-frame master dark image (dark current), badpixel-map, rms-map and rms-map.

Name	Type	Description
ZPL_ MASTER_ DARK_ IMAGING_ CAM2	FITS[Im(8)]	The resulting master dark frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions (2 master frames), grouped by the following order: - odd sub-frame master dark image (informative), badpixel-map, ncomb-map and rms-map; - even sub-frame master dark image (dark current), badpixel-map, ncomb-map and rms-map.

10.7 sph_zpl_intensity_flat

Purpose:

Create intensity flat field, polarimetric modes.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ INT_ FLAT_ FIELD_ RAW	Raw data	Yes	0	Any
ZPL_ INT_ FLAT_ PREPROC	Calibration	Yes	0	Any
ZPL_ INT_ FLAT_ PREPROC_ CAM1	Calibration	Yes	0	Any
ZPL_ INT_ FLAT_ PREPROC_ CAM2	Calibration	Yes	0	Any
ZPL_ MASTER_ BIAS	Calibration	Yes	0	1
ZPL_ MASTER_ DARK	Calibration	Yes	0	1
ZPL_ MASTER_ BIAS_ CAM1	Calibration	Yes	0	1
ZPL_ MASTER_ BIAS_ CAM2	Calibration	Yes	0	1
ZPL_ MASTER_ DARK_ CAM1	Calibration	Yes	0	1
ZPL_ MASTER_ DARK_ CAM2	Calibration	Yes	0	1
ZPL_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if the pre-processed data are used. As the format of the zimpol pre-processed data is complicated, this keyword was introduced in order to garantee that the pre-processed input frames are polarimetric pre-processed data, produced by the sph_zpl_preproc utility recipe. Note: if raw data are used (default), then all keywords needed for the pre-processing recipe (see sph_zpl_preproc) must be presented in the raw data.

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.master_ intensity_ flat.outfilename	string	The output filename of the final iff product for the camera-1/2. This product is usually used in all subsequent polarimetric recipes. Please also see the esorex documentation for naming of output products.	zpl_ master_ intensity_ flat.fits	-
zpl.intensity_ flat.outfilename	string	The output filename for the quad image iff product for the camera-2. Please also see the esorex documentation for naming of output products.	zpl_ quad_ intensity_ flat.fits	-
zpl.master_ intensity_ flat.outfilename_ cam1	string	The output filename of the final iff product for the camera-1/2. This product is usually used in all subsequent polarimetric recipes. Please also see the esorex documentation for naming of output products.	zpl_ master_ intensity_ flat_ cam1.fits	-
zpl.intensity_ flat.outfilename_ cam1	string	The output filename of the quad image iff product of the camera-1. Please also see the esorex documentation for naming of output products.	zpl_ quad_ intensity_ flat_ cam1.fits	-
zpl.master_ intensity_ flat.outfilename_ cam2	string	The output filename of the final iff product for the camera-2. This product is usually used in all subsequent polarimetric recipes. Please also see the esorex documentation for naming of output products.	zpl_ master_ intensity_ flat_ cam2.fits	-
zpl.intensity_ flat.outfilename_ cam2	string	The output filename for the quad image iff product for the camera-2. Please also see the esorex documentation for naming of output products.	zpl_ quad_ intensity_ flat_ cam2.fits	-
zpl.intensity_ flat.subtract_ overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.intensity_ flat.badpixfilename	string	Controls the filename of the badpixel map, if requested for output. Ignored if make_badpix is FALSE.	zpl_ intensity_ flat_ nonlin_ badpixels.fits	-
zpl.intensity_ flat.badpixfilename_ cam1	string	Controls the filename of the badpixel map, if requested for output. Ignored if make_badpix is FALSE.	zpl_ intensity_ flat_ nonlin_ badpixels_ cam1.fits	-
zpl.intensity_ flat.badpixfilename_ cam2	string	Controls the filename of the badpixel map, if requested for output. Ignored if make_badpix is FALSE.	zpl_ intensity_ flat_ nonlin_ badpixels_ cam2.fits	-

Name	Type	Description	Default	Allowed vals.
zpl.intensity_ flat.robust_ fit	bool	Controls if fitting method is to be a robust linear fit. This will reduce the effect of cosmic rays and other temporary bad pixels. See e.g. Numerical Recipes for a description of the algorithm	0	-
zpl.intensity_ flat.collapse	bool	Controls if the collapse is used to calculate intensity flat instead of the linear fitting	1	-
zpl.intensity_ flat.coll_ alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median, 2 = Clean Mean. Default is 2 = Clean Mean	2	0,1,2
zpl.intensity_ flat.coll_ alg.clean_ mean.reject_ high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.intensity_ flat.coll_ alg.clean_ mean.reject_ low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.intensity_ flat.sigma_ clip	double	The sigma clipping value for static badpixel detection. Default is 5.	5.0	0.0-200.0
zpl.intensity_ flat.badpix_ lowtolerance	double	Threshold value for linearity badpixels. All pixels that have a flat field (slope) below this value will be flagged as bad.	0.1	-
zpl.intensity_ flat.badpix_ uptolerance	double	Threshold value for linearity badpixels. All pixels that have a flat field (slope) above this value will be flagged as bad.	10.0	-
zpl.intensity_ flat.badpix_ chisqtolerance	double	Threshold value for linearity badpixels. All pixels that have chi-squared value for the linear fit that is above this value will be flagged as bad.	50.0	-
zpl.intensity_ flat.quadimage_ weight_ mean	bool	Controls if the combining of the collapsed quad image to the final single master frame product is carried out using weghted mean or standard mean.	0	-
zpl.intensity_ flat.keep_ intermediate	bool	Flag to set if intermediate date must be saved, namely pre-processed and overscan pre-processed subtracted data, linbadpix map and non-normalized products (FALSE)	0	-
zpl.preproc.outfilename_ cam1	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-1.	preproc_ cam1	-
zpl.preproc.outfilename_ cam2	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-2.	preproc_ cam2	-

Description:

This recipe creates the master intensity flat field calibration framefor the polarization modes. The input frames might be either intesity flat raw frames with the ZPL_INT_FLAT_FIELD_RAW tag or pre-processed intensity flat frames, which should carry the ZPL_INT_FLAT_PREPROC_CAM1 and/or ZPL_INT_FLAT_PREPROC_CAM2 tags, and master bias frames (if any) with the ZPL_MASTER_BIAS_CAM1 and/or ZPL_MASTER_BIAS_CAM2 tags, and master dark frames (if any) with the ZPL_MASTER_DARK_CAM1 and/or ZPL_MASTER_DARK_CAM2 tags.If input frames are raw frames then the intensity flat recipe first performs the pre-processing step for all input frames (raw cubes),creating corresponding pre-processed frames (cubes) for both ZIMPOL cameras (see also sph_zpl_preproc for the detailed description of the pre-processing step). The recipe creates for both cameras the intensity flat field calibration frame, using the input exposures which should be taken as described in the zimpol calibration plan.There are two main different methods to calculate the master intensity flatfield:

- combining frames (plus normalizing): in this case, the raw frames must be acquired with the same DIT and filter;
- linear fitting method of the individual pixels: in this case, the raw frames must be acquired either with a different DIT or with a different intensity of the lamp, but with the same filter.

The first <combining frames method> combines pre-processed raw intensity flatfield frame (= all planes) from the pre-processed cube(s) using the specified collapse algorithm (usually the clean_mean algorithm, defined as a default one).After all pre-processed frames (all 4 zimpol exposure sub-frames) are combined in this way,the badpixel maps are determined on the results, using a simple sigma clipping algorithm.It sets the bad/hot pixels to be all those that are further than the >>specified sigma x the total RMS<< of the whole image away from the image median. Note that the badpixels which are stored in the master flat field product itself will contain all badpixels,accumulated at this point in the cascade (i.e. badpixels from the intensity flat and master dark, and master bias, if exists). The quad image intensity flat field products for both cameras are then written out in the fits filesin the QUAD IMAGE format:

1. zpl exp phase zero odd sub-frame:
   - intensity flat field image,
   - badpixel-map,
   - ncomb-map,
   - rms-map;
2. zpl exp phase zero even sub-frame:
   - intensity flat field image
   - badpixel-map
   - ncomb-map
   - rms-map;
3. zpl exp phase PI odd sub-frame:
   - intensity flat field image,
   - badpixel-map,
   - ncomb-map,
   - rms-map;

4. zpl exp phase PI even sub-frame:
   - intensity flat field image,
   - badpixel-map,
   - ncomb-map,
   - rms-map.

Another products of the recipe are saved for both cameras in the MASTER FRAME format after combination of the resulting quad image into the final master frame products. This can be done either by simple averaging of the four sub-frames of the quad image or by using weighted mean formula where rms-map (calculated from error propagation) is taking into account to produce needed weights. These master intensity flat field calibration products (in the format of the MASTER FRAME) for both cameras are usually used in all subsequent polarimetric recipes that need to remove the pixel to pixel variation in the signal response of the detector. However, the quad image intensity flat field (in the format of the QUAD IMAGE, considered for monitoring purposes), may also feed the subsequent polarimetric recipes.The second <linear fitting flat fielding procedure> descibed below (identical to that for the IFS and IRDIS) is thenapplied to the each zpl exp polarimetric sub-frames (zero odd, zero even, pi odd, pi even) seperately.

1. The mean value is determined for the respective sub-frame forall exposures.
2. For every pixel $p = (x, y)$, a set of m_i,$v_i(x, y)$ data pairs are stored with m_i being the exposure mean value and $v_i(x, y)$ being the pixel value for exposure i.
3. The flat field value is defined as the slope c_i of a linear fit F to the datam_i,v_i.
4. The fit itself is performed either using a maximum likelyhood methodor a robust fitting method which minimizes the sum of the absolute value of the deviations rather than the sum of the squares of the deviations (see e.g. Numerical Recipes for the algorithm). The robust fitting method will yield better results when significant outliers (e.g. due to cosmic rays) canbe expected.
5. The flat field values (linear coefficients) are saved as an image as the main product of the recipe in the same QUAD IMAGE format (see above).

Aditionally, the recipe may also produce a separate output of all pixels that are identified as non-linear. The criteria for non-linearity are set by the userparameters and can be either pixels that have a flat field value outside specified bounds and/or pixels for which the linear fit produces a reduced chi-squared abovea given threshold value. For reliable non-linearity flagging using the reduced chi-squared it is necessary to use many high quality input exposures.Since the badpixel treatment is somewhat complicated, some important points: the badpixelsthat are stored in the master flat field itself as produced by this recipe contain all the badpixels (for each sub-framesindividually) at this point in the cascade. Pixels thatwere marked as bad from the input static badpixel map are also marked as bad here.The optional static badpixel output that is produced contains strictly only those pixel that theflat field recipe itself deemed to be bad. This does not necessarily include all the badpixelsfrom the static badpixel input file.

Products:

Name	Type	Description

Name	Type	Description
ZPL_ INT_ FLAT_ FIELD_ MASTER	FITS[Im(4)]	The final master intensity flad field frame of the MASTER FRAME format which is used in all subsequent polarimetric recipes. This frame contains 4 image extensions: combined intensity flat image, badpixel-map, map of yhe combined number frames and rms.
ZPL_ INT_ FLAT_ FIELD	FITS[Im(16)]	The resulting master intensity flad field frame of the QUAD IMAGE format. This frame contains 16 image extensions: intensity flat field zero odd image, badpixel-map, rms-map and weight-map; intensity flat field zero even image, badpixel-map, rms-map and weight-map; intensity flat field pi odd image, badpixel-map, rms-map and weight-map; intensity flat field pi even image, badpixel-map, rms-map and weight-map.
ZPL_ INT_ FLAT_ FIELD_ MASTER_ CAM1	FITS[Im(4)]	The final master intensity flad field frame of the MASTER FRAME format which is used in all subsequent polarimetric recipes. This frame contains 4 image extensions: combined intensity flat image, badpixel-map, map of yhe combined number frames and rms.
ZPL_ INT_ FLAT_ FIELD_ MASTER_ CAM2	FITS[Im(4)]	The final master intensity flad field frame of the MASTER FRAME format which is used in in all subsequent polarimetric recipes. This frame contains 4 image extensions: combined intensity flat image, badpixel-map, map of yhe combined number frames and rms.
ZPL_ INT_ FLAT_ FIELD_ CAM1	FITS[Im(16)]	The resulting master intensity flad field frame of the QUAD IMAGE format. This frame contains 16 image extensions: intensity flat field zero odd image, badpixel-map, rms-map and weight-map; intensity flat field zero even image, badpixel-map, rms-map and weight-map; intensity flat field pi odd image, badpixel-map, rms-map and weight-map; intensity flat field pi even image, badpixel-map, rms-map and weight-map.

Name	Type	Description
ZPL_ INT_ FLAT_ FIELD_ CAM2	FITS[Im(16)]	The resulting master intensity flad field frame of the QUAD IMAGE format. This frame contains 16 image extensions: intensity flat field zero odd image, badpixel-map, rms-map and weight-map; intensity flat field zero even image, badpixel-map, rms-map and weight-map; intensity flat field pi odd image, badpixel-map, rms-map and weight-map; intensity flat field pi even image, badpixel-map, rms-map and weight-map.
ZPL_ NON_ LINEAR_ BADPIXELMAP	FITS[Im(4)]	Optional output of all the non-linear pixels determined. All pixels as determined in this recipe using the zpl.intensity_flat.badpix_low(up)tolerance parameters. phase zero odd sub-frame image; phase zero even sub-frame image; phase PI odd sub-frame image; phase PI even sub-frame.
ZPL_ NON_ LINEAR_ BADPIXELMAP_ CAM1	FITS[Im(4)]	Optional output of all the non-linear pixels determined. All pixels as determined in this recipe using the zpl.intensity_flat.badpix_low(up)tolerance parameters. phase zero odd sub-frame image; phase zero even sub-frame image; phase PI odd sub-frame image; phase PI even sub-frame.
ZPL_ NON_ LINEAR_ BADPIXELMAP_ CAM2	FITS[Im(4)]	Optional output of all the non-linear pixels determined. All pixels as determined in this recipe using the zpl.intensity_flat.badpix_low(up)tolerance parameters. phase zero odd sub-frame image; phase zero even sub-frame image; phase PI odd sub-frame image; phase PI even sub-frame.

10.8 sph_zpl_intensity_flat_imaging

Purpose:

Create intensity flat field, imaging mode.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_INT_FLAT_FIELD_IMAGING_RAW	Raw data	Yes	0	Any
ZPL_INT_FLAT_FIELD_IMAGING_PREPROC_RAW	Raw data	Yes	0	Any
ZPL_INT_FLAT_FIELD_IMAGING_PREPROC_CAM1	Calibration	Yes	0	Any
ZPL_INT_FLAT_FIELD_IMAGING_PREPROC_CAM2	Calibration	Yes	0	Any
ZPL_MASTER_BIAS_IMAGING	Calibration	Yes	0	1
ZPL_MASTER_BIAS_IMAGING_CAM1	Calibration	Yes	0	1
ZPL_MASTER_BIAS_IMAGING_CAM2	Calibration	Yes	0	1
ZPL_MASTER_DARK_IMAGING	Calibration	Yes	0	1
ZPL_MASTER_DARK_IMAGING_CAM1	Calibration	Yes	0	1
ZPL_MASTER_DARK_IMAGING_CAM2	Calibration	Yes	0	1
ZPL_STATIC_BADPIXELMAP_IMAGING	Calibration	Yes	0	1
ZPL_STATIC_BADPIXELMAP_IMAGING_CAM1	Calibration	Yes	0	1
ZPL_STATIC_BADPIXELMAP_IMAGING_CAM2	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if the pre-processed data are used. As the format of the zimpol data is complicated, this keyword is introduced in order to garantee that the input frames are imaging pre-processed data, produced by the sph_zpl_preproc_imaging recipe which added this keyword automatically. The value of this keyword is set up to >>SPH PC PREPROC ZPL EXP IMAGING<<.

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.intensity_flat_imaging.outfilename	string	The output filename for the product, camera-1/2. Please also see the esorex documentation for naming of output products.	zpl_intensity_flat_imaging.fits	-
zpl.intensity_flat_imaging.outfilename_cam1	string	The output filename for the product, camera-1. Please also see the esorex documentation for naming of output products.	zpl_intensity_flat_imaging_cam1.fits	-
zpl.intensity_flat_imaging.outfilename_cam2	string	The output filename for the product, camera-2. Please also see the esorex documentation for naming of output products.	zpl_intensity_flat_imaging_cam2.fits	-

Name	Type	Description	Default	Allowed vals.
zpl.intensity_ flat_ imaging.subtract_ overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.intensity_ flat_ imaging.badpixfilename	string	Controls the filename of the badpixel map, if requested for output. Ignored if make_badpix is FALSE.	zpl_ intensity_ flat_ imaging_ nonlin_ badpixels.fits	-
zpl.intensity_ flat_ imag-ing.badpixfilename_ cam1	string	Controls the filename of the badpixel map, if requested for output. Ignored if make_badpix is FALSE.	zpl_ intensity_ flat_ imaging_ nonlin_ badpixels_ cam1.fits	-
zpl.intensity_ flat_ imag-ing.badpixfilename_ cam2	string	Controls the filename of the badpixel map, if requested for output. Ignored if make_badpix is FALSE.	zpl_ intensity_ flat_ imaging_ nonlin_ badpixels_ cam2.fits	-
zpl.intensity_ flat_ imaging.robust_ fit	bool	Controls if fitting method is to be a robust linear fit. This will reduce the effect of cosmic rays and other temporary bad pixels. See e.g. Numerical Recipes for a description of the algorithm	0	-
zpl.intensity_ flat_ imaging.collapse	bool	Controls if the collapse is used to calculate intensity flat instead of the fitting	1	-
zpl.intensity_ flat_ imaging.coll_ alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median, 2 = Clean Mean. Default is 2 = Clean Mean	2	0,1,2
zpl.intensity_ flat_ imaging.coll_ alg.clean_ mean.reject_ high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.intensity_ flat_ imaging.coll_ alg.clean_ mean.reject_ low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.intensity_ flat_ imaging.sigma_ clip	double	The sigma clipping value for static badpixel detection. Default is 5.	5.0	0.0-200.0
zpl.intensity_ flat_ imaging.badpix_ lowtolerance	double	Threshold value for linearity badpixels. All pixels that have a flat field (slope) below this value will be flagged as bad.	0.1	-
zpl.intensity_ flat_ imaging.badpix_ uptolerance	double	Threshold value for linearity badpixels. All pixels that have a flat field (slope) above this value will be flagged as bad.	10.0	-
zpl.intensity_ flat_ imaging.badpix_ chisqtolerance	double	Threshold value for linearity badpixels. All pixels that have chi-squared value for the linear fit that is above this value will be flagged as bad	50.0	-

Name	Type	Description	Default	Allowed vals.
zpl.intensity_flat_imaging.keep_intermediate	bool	Flag to set if intermediate date must be saved, namely pre-processed and overscan pre-processed subtracted data, linbadpix map and non-normalized products (FALSE)	0	-
zpl.preproc.outfilename_cam1	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-1.	preproc_cam1	-
zpl.preproc.outfilename_cam2	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-2.	preproc_cam2	-

Description:

The recipe creates the intensity flat field calibration frames for the imaging modes.The input frames might be either intensity flat raw frames with the ZPL_INT_FLAT_IMAGING_RAW tag or pre-processed intensity flat frames, which should carry the ZPL_INT_FLAT_IMAGING_PREPROC_CAM1 and/or ZPL_INT_FLAT_IMAGING_PREPROC_CAM2 tags, and master bias frames (if any) with the ZPL_MASTER_BIAS_IMAGING_CAM1 and/or ZPL_MASTER_BIAS_IMAGING_CAM2 tags, and and master dark frames (if any) with the ZPL_MASTER_DARK_IMAGING_CAM1 and/or ZPL_MASTER_DARK_IMAGING_CAM2 tags.If input frames are raw frames then the intensity flat field recipe first performs the pre-processing step for all input frames (raw cubes),creating corresponding pre-processed frames (cubes) for both ZIMPOL cameras (see also sph_zpl_preproc_imaging for the detailed description of the pre-processing step). There are two main different methods to calculate the master intensity flatfield:

- combining frames (plus normalizing): in this case, the raw frames must be acquired with the same DIT and filter;
- linear fitting method of the individual pixels: in this case, the raw frames must be acquired either with a different DIT or with a different intensity of the lamp, but with the same filter.

The first <combining frames method> combines pre-processed raw intensity flatfield frame (= all planes) from the pre-processed cube(s) using the specified collapse algorithm (usually the clean_mean algorithm, defined as a default one).After all pre-processed frames (all 4 zimpol exposure sub-frames) are combined in this way,the badpixel maps are determined on the results, using a simple sigma clipping algorithm.It sets the bad/hot pixels to be all those that are further than the >>specified sigma x the total RMS<< of the whole image away from the image median. Note that the badpixels that are stored in the master flat field itself as produced by this recipe contain all the badpixels (for each sub-framesindividually) at this point in the cascade (i.e. badpixels from the master dark and master bias, if exists). The resulting master intensity flat field products for both cameras are then written out in the fits files in the DOUBLE IMAGE format:

1. zpl exp imaging odd sub-frame (informative component):
    - intensity flat field image,
    - badpixel-map,
    - ncomb-map,
    - rms-map;

2. zpl exp imaging even sub-frame (dark current component):
   - intensity flat field image
   - badpixel-map
   - ncomb-map
   - rms-map;

The second <linear fitting flat fielding procedure> descibed below (identical to that for the IFS and IRDIS) is thenapplied to the each >>zpl exp imaging sub-frames<< (odd - informative component, even-dark current component) seperately.

1. The mean value is determined for the respective sub-frame forall exposures.
2. For every pixel $p = (x, y)$, a set of m_i,$v_i(x, y)$ data pairs are stored with m_i being the exposure mean value and $v_i(x, y)$ being the pixel value for exposure i.
3. The flat field value is defined as the slope c_i of a linear fit F to the datam_i,v_i.
4. The fit itself is performed either using a maximum likelyhood methodor a robust fitting method which minimizes the sum of the absolute value of the deviations rather than the sum of the squares of the deviations (see e.g. Numerical Recipes for the algorithm). The robust fitting method will yield better results when significant outliers (e.g. due to cosmic rays) canbe expected.
5. The flat field values (linear coefficients) are saved as an image as the main product of the recipe in the same DOUBLE IMAGE format (see above).

Aditionally, the recipe may also produce a separate output of all pixels that are identified as non-linear. The criteria for non-linearity are set by the userparameters and can be either pixels that have a flat field value outside specified bounds and/or pixels for which the linear fit produces a reduced chi-squared abovea given threshold value. For reliable non-linearity flagging using the reduced chi-squared it is necessary to use many high quality input exposures.Since the badpixel treatment is somewhat complicated, some important points: the badpixelsthat are stored in the master flat field itself as produced by this recipe contain all the badpixels (for each subframesindividually) at this point in the cascade. Pixels thatwere marked as bad from the input static badpixel map are also marked as bad here.The optional static badpixel output that is produced contains strictly only those pixel that theflat field recipe itself deemed to be bad. This does not necessarily include all the badpixelsfrom the static badpixel input file.The intensity flat field calibration products for both cameras my be used in all subsequent imaging mode recipes if one needs to remove the pixel to pixel variation of the signal response on the detector.

Products:

Name	Type	Description
ZPL_ INT_ FLAT_ FIELD_ IMAGING	FITS[Im(8)]	The resulting intensity imaging flat field frame is of the DOUBLE IMAGE format. This DOUBLE IMAGE frame contains 8 image extensions (2 master frames), grouped by the following: odd sub-frame intensity flat field image (informative), badpixel-map, rms-map and weight-map; even sub-frame intensity flat field image (dark current), badpixel-map, rms-map and weight-map.

Name	Type	Description
ZPL_ INT_ FLAT_ FIELD_ IMAGING_ CAM1	FITS[Im(8)]	The resulting intensity imaging flat field frame is of the DOUBLE IMAGE format. This DOUBLE IMAGE frame contains 8 image extensions (2 master frames), grouped by the following: odd sub-frame intensity flat field image (informative), badpixel-map, rms-map and weight-map; even sub-frame intensity flat field image (dark current), badpixel-map, rms-map and weight-map.
ZPL_ INT_ FLAT_ FIELD_ IMAGING_ CAM2	FITS[Im(8)]	The resulting intensity imaging flat field frame is of the DOUBLE IMAGE format. This DOUBLE IMAGE frame contains 8 image extensions (2 master frames), grouped by the following: odd sub-frame intensity flat field image (informative), badpixel-map, rms-map and weight-map; even sub-frame intensity flat field image (dark current), badpixel-map, rms-map and weight-map.
ZPL_ NON_ LINEAR_ BADPIXELMAP_ IMAGING	FITS[Im(4)]	Optional output of all the non-linear pixels determined. All pixels as determined in this recipe using the zpl.intensity_flat.badpix_low(up)tolerance parameters. This badpixel map frame is of ZPL EXP IMAGING format: odd sub-frame image; even sub-frame image (dark current).
ZPL_ NON_ LINEAR_ BADPIXELMAP_ IMAGING_ CAM1	FITS[Im(4)]	Optional output of all the non-linear pixels determined. All pixels as determined in this recipe using the zpl.intensity_flat.badpix_low(up)tolerance parameters. This badpixel map frame is of ZPL EXP IMAGING format: odd sub-frame image; even sub-frame image (dark current).
ZPL_ NON_ LINEAR_ BADPIXELMAP_ IMAGING_ CAM2	FITS[Im(4)]	Optional output of all the non-linear pixels determined. All pixels as determined in this recipe using the zpl.intensity_flat.badpix_low(up)tolerance parameters. This badpixel map frame is of ZPL EXP IMAGING format: odd sub-frame image; even sub-frame image (dark current).

10.9 sph_zpl_polarization_flat

Purpose:

Create polarization flat field, polarimetric modes.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ POL_ FLAT_ FIELD_ RAW	Raw data	Yes	0	Any
ZPL_ POL_ FLAT_ PREPROC	Calibration	Yes	0	Any
ZPL_ POL_ FLAT_ PREPROC_ CAM1	Calibration	Yes	0	Any
ZPL_ POL_ FLAT_ PREPROC_ CAM2	Calibration	Yes	0	Any
ZPL_ MASTER_ BIAS	Calibration	Yes	0	1
ZPL_ MASTER_ BIAS_ CAM1	Calibration	Yes	0	1
ZPL_ MASTER_ BIAS_ CAM2	Calibration	Yes	0	1
ZPL_ MASTER_ DARK	Calibration	Yes	0	1
ZPL_ MASTER_ DARK_ CAM1	Calibration	Yes	0	1
ZPL_ MASTER_ DARK_ CAM2	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ CAM1	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ CAM2	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ MASTER	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ MASTER_ CAM1	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ MASTER_ CAM2	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if the pre-processed data are used. As the format of the zimpol pre-processed data is complicated, this keyword was introduced in order to garantee that the pre-processed input frames are polarimetric pre-processed data, produced by the sph_zpl_preproc utility recipe. Note: if raw data are used (default), then all keywords needed for the pre-processing recipe (see sph_zpl_preproc) must be presented in the raw data.

Parameters:

Name	Type	Description	Default	Allowed vals.

Name	Type	Description	Default	Allowed vals.
zpl.polarization_ flat.outfilename	string	The output filename for the product for the camera-1/2. Please also see the esorex documentation for naming of output products.	zpl_ polarization_ flat.fits	-
zpl.polarization_ flat.outfilename_ cam1	string	The output filename for the product for the camera-1. Please also see the esorex documentation for naming of output products.	zpl_ polarization_ flat_ cam1.fits	-
zpl.polarization_ flat.outfilename_ cam2	string	The output filename for the product for the camera-2. Please also see the esorex documentation for naming of output products.	zpl_ polarization_ flat_ cam2.fits	-
zpl.polarization_ flat.subtract_ overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.polarization_ flat.coll_ alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median, 2 = Clean Mean. Default is 2 = Clean Mean	2	0,1,2
zpl.polarization_ flat.coll_ alg.clean_ mean.reject_ high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.polarization_ flat.coll_ alg.clean_ mean.reject_ low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.polarization_ flat.keep_ intermediate	bool	Flag to set if intermediate date must be saved, namely pre-processed and overscan pre-processed subtracted data (FALSE)	0	-
zpl.preproc.outfilename_ cam1	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-1.	preproc_ cam1.fits	-
zpl.preproc.outfilename_ cam2	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-2.	preproc_ cam2.fits	-

Description:

This recipe creates the polarization flat field calibration frame for both cameras.The input frames might be either intesity flat raw frames with the ZPL_POL_FLAT_FIELD_RAW tag or pre-processed polarization flatframes, which should carry the ZPL_POL_FLAT_PREPROC_CAM1 and/or tag ZPL_POL_FLAT_PREPROC_CAM2 tags, andmaster bias frames (if any) with the ZPL_MASTER_BIAS_CAM1 and/or ZPL_MASTER_BIAS_CAM2 tags, and master dark frames (if any) with the ZPL_MASTER_DARK_CAM1 and/or ZPL_MASTER_DARK_CAM2 tags, andmaster intensity flat field calibration frames with the ZPL_INT_FLAT_FIELD_MASTER_CAM1 and/or ZPL_INT_FLAT_FIELD_MASTER_CAM2 tags.The intensity flat calibration frames can be also used in the format of the QUAD IMAGE (see the description in sph_zpl_intensity_flat recipe) with the corresponding ZPL_INT_FLAT_FIELD_CAM1 and/or ZPL_INT_FLAT_FIELD_CAM2 tags. If both formats of the intensity flat field calibrations are presented in sof-file the MASTER format will be used.If input frames are raw frames then the polarization flat recipe first performs the pre-processing step for all input frames (raw cubes),creating corresponding pre-processed frames (cubes) for both ZIMPOL cameras (see also sph_zpl_preproc for the detailed description of the pre-processing step). Then, all the pre-processed frames are read and combined using the specified collapse algorithm (usually the clean_mean algorithm, defined as a default one) for each zpl exposure sub-frames.The combined frames for both cameras are of the QUAD IMAGE (16 extensions) format specified as follows:- zpl exp phase zero odd sub-frame combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase zero even sub-frame combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase PI odd sub-frame master combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase PI even sub-frame master combined image, badpixel-map, ncomb-map and rms-map.The master bias, dark and intensity flat field are applied to this combined master frame andthen the stock parameters (I,P) are calculated. The output master polarization flat field is writtenout in the DOUBLE IMAGE (8 images) format specified as follows: - master intensity stock parameter image, badpixel-map, ncomb-map and rms-map;- master polarization stock parameter image, badpixel-map, ncomb-map and rms-map.The master polarization flat field products for both cameras are used in all subsequent polarization recipes.

Products:

Name	Type	Description
ZPL_ POL_ FLAT_ FIELD	FITS[Im(8)]	The resulting polarization flat filed frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions grouped by the following order: master intensity image, badpixel-map, ncomb-map and rms-map; master polarization image, badpixel-map, ncomb-map and rms-map.
ZPL_ POL_ FLAT_ FIELD_ CAM1	FITS[Im(8)]	The resulting polarization flat filed frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions grouped by the following order: master intensity image, badpixel-map, ncomb-map and rms-map; master polarization image, badpixel-map, ncomb-map and rms-map.
ZPL_ POL_ FLAT_ FIELD_ CAM2	FITS[Im(8)]	The resulting polarization flat filed frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions grouped by the following order: intensity image, badpixel-map, ncomb-map and rms-map; polarization image, badpixel-map, ncomb-map and rms-map.

10.10 sph_zpl_modem_efficiency

Purpose:

Create modem efficiency, polarimetric modes.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_MODEM_EFF_RAW	Raw data	Yes	0	Any
ZPL_MODEM_EFF_PREPROC_RAW	Raw data	Yes	0	Any
ZPL_MODEM_EFF_PREPROC_CAM1	Calibration	Yes	0	Any
ZPL_MODEM_EFF_PREPROC_CAM2	Calibration	Yes	0	Any
ZPL_MASTER_BIAS	Calibration	Yes	0	1
ZPL_MASTER_BIAS_CAM1	Calibration	Yes	0	1
ZPL_MASTER_BIAS_CAM2	Calibration	Yes	0	1
ZPL_MASTER_DARK	Calibration	Yes	0	1
ZPL_MASTER_DARK_CAM1	Calibration	Yes	0	1
ZPL_MASTER_DARK_CAM2	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD_CAM1	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD_CAM2	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD_MASTER	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD_MASTER_CAM1	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD_MASTER_CAM2	Calibration	Yes	0	1
ZPL_POL_FLAT_FIELD	Calibration	Yes	0	1
ZPL_POL_FLAT_FIELD_CAM1	Calibration	Yes	0	1
ZPL_POL_FLAT_FIELD_CAM2	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if the pre-processed data are used. As the format of the zimpol pre-processed data is complicated, this keyword was introduced in order to garantee that the pre-processed input frames are polarimetric pre-processed data, produced by the sph_zpl_preproc utility recipe. Note: if raw data are used (default), then all keywords needed for the pre-processing recipe (see sph_zpl_preproc) must be presented in the raw data.
ESO OCS3 ZIMPOL POL STOKES	string	No	Stokes parameters (Qplus, Qminus)

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.modem_ efficiency.outfilename	string	The output filename of the final modem efficiency product for the camera-1/2. This product is used in all subsequent polarimetric recipes. Please also see the esorex documentation for naming of output products.	zpl_ modem_ efficiency.fits	-
zpl.modem_ efficiency.outfilename_ cam1	string	The output filename of the final modem efficiency product for the camera-1. This product is used in all subsequent polarimetric recipes. Please also see the esorex documentation for naming of output products.	zpl_ modem_ efficiency_ cam1.fits	-
zpl.modem_ efficiency.outfilename_ cam2	string	The output filename of the final modem efficiency product for the camera-2. This product is used in all subsequent polarimetric recipes. Please also see the esorex documentation for naming of output products.	zpl_ modem_ efficiency_ cam2.fits	-
zpl.modem_ efficiency_ qplus.outfilename	string	The output filename of the qplus modem efficiency product for the camera-1/2. Please also see the esorex documentation for naming of output products.	zpl_ modem_ efficiency_ plus.fits	-
zpl.modem_ efficiency_ qplus.outfilename_ cam1	string	The output filename of the qplus modem efficiency product for the camera-1. Please also see the esorex documentation for naming of output products.	zpl_ modem_ efficiency_ plus_ cam1.fits	-
zpl.modem_ efficiency_ qminus.outfilename	string	The output filename of the qminus modem efficiency product for the camera-1/2. Please also see the esorex documentation for naming of output products.	zpl_ modem_ efficiency_ minus.fits	-
zpl.modem_ efficiency_ qminus.outfilename_ cam1	string	The output filename of the qminus modem efficiency product for the camera-1. Please also see the esorex documentation for naming of output products.	zpl_ modem_ efficiency_ minus_ cam1.fits	-
zpl.modem_ efficiency_ qplus.outfilename_ cam2	string	The output filename of the qplus modem efficiency product for the camera-2. Please also see the esorex documentation for naming of output products.	zpl_ modem_ efficiency_ plus_ cam2.fits	-
zpl.modem_ efficiency_ qminus.outfilename_ cam2	string	The output filename of the qminus modem efficiency product for the camera-2. Please also see the esorex documentation for naming of output products.	zpl_ modem_ efficiency_ minus_ cam2.fits	-

Name	Type	Description	Default	Allowed vals.
zpl.modem_ efficiency.subtract_ overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.modem_ efficiency.coll_ alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median, 2 = Clean Mean. Default is 2 = Clean Mean	2	0,1,2
zpl.modem_ efficiency.coll_ alg.clean_ mean.reject_ high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.modem_ efficiency.coll_ alg.clean_ mean.reject_ low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.modem_ efficiency.keep_ intermediate	bool	Flag to set if intermediate date must be saved, namely pre-processed and overscan pre-processed subtracted data (FALSE)	0	-
zpl.preproc.outfilename_ cam1	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-1.	preproc_ cam1	-
zpl.preproc.outfilename_ cam2	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-2.	preproc_ cam2	-

Description:

The recipe creates master modulation/demodulation(modem) efficiency calibration product, using the input exposures which should be taken as described in the calibration plan. The input frames might be either modem efficiency raw frames with the ZPL_MODEM_EFF_RAW tag,or pre-processed modem efficiency frames, which should carry the ZPL_MODEM_EFF_PREPROC_CAM1 and/or ZPL_MODEM_EFF_PREPROC_CAM2 tags, andmaster bias calibration frames (if any) with the ZPL_MASTER_BIAS_CAM1 and/or ZPL_MASTER_BIAS_CAM2 tags, and master dark calibration frames (if any) with the ZPL_MASTER_DARK_CAM1 and/or ZPL_MASTER_DARK_CAM2 tags, andmaster intensity flat field calibration frames with the ZPL_INT_FLAT_FIELD_MASTER_CAM1 and/or ZPL_INT_FLAT_FIELD_MASTER_CAM2 tags, andpolariztion flat field calibration frames with the ZPL_POL_FLAT_PREPROC_CAM1 and/or ZPL_POL_FLAT_PREPROC_CAM2 tags.The intensity flat calibration frames can be also used in the format of the QUAD IMAGE (see the description in sph_zpl_intensity_flat recipe) with the corresponding ZPL_INT_FLAT_FIELD_CAM1 and/or ZPL_INT_FLAT_FIELD_CAM2 tags. If both formats of the intensity flat field calibrations are presented in sof-file the MASTER format will be used. If input frames are raw frames then the recipe first performs the pre-processing step for all input frames (raw cubes),creating corresponding pre-processed frames (cubes) for both ZIMPOL cameras (see also sph_zpl_preproc for the detailed description of the pre-processing step). Then all pre-processed modem frames are organized in the two groups distinguished from each other by the opposite sign of the stock parameter [Qplus, Qminus]. The frames from each group are combined using the specified collapse algorithm (usually the clean_mean algorithm, defined as a default one) for each zpl exposure sub-frames.The combined frame is of the QUAD IMAGE (16 extensions) format specified as follows:- zpl exp phase zero odd sub-frame combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase zero even sub-frame combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase PI odd sub-frame master combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase PI even sub-frame master combined image, badpixel-map, ncomb-map and rms-map.The master bias, dark and intensity flat field are applied to the two [Qplus, Qminus] combined master frames and then the stock parameters (I,P) are calculated for both frames and both cameras in the form of DOUBLE IMAGE: - master intensity stock parameter image, badpixel-map, ncomb-map and rms-map; - master polarization stock parameter image, badpixel-map, ncomb-map and rms-map.Then, the polarization flat field is applied to the [Qplus, Qminus] stock parameters double image frames.This intermediate modem efficiency products for +Q and -Q are saved in the separate files (quality check). Finally, the two opposite polarization frames are combined by subtracting the MINUS polarization image frame from the PLUS one. The output modem polarization efficiency frames for both camera are calculated by dividing the polarization image by the intensity image (P/I). The final modem products are thus of the MASTER FRAME format specified as follows:modem efficiency image, badpixel-map, ncomb-map and rms-map.Note: if rawdata consist only of Qplus-data (or only Qminus-data) then the final products will be created directly from Qplus (or Qminus)double image (P/I). The final modem efficiency products for both cameras are used in all subsequent polarization recipes.

Products:

Name	Type	Description
ZPL_ MODEM_ EFF	FITS[Im(4)]	The final modem efficiency frame is of the MASTER FRAME format. This frame contains 4 image extensions: modem efficiency image, badpixel-map, ncomb-map and rms-map.
ZPL_ MODEM_ EFF_ CAM1	FITS[Im(4)]	The final modem efficiency frame is of the MASTER FRAME format. This frame contains 4 image extensions: modem efficiency image, badpixel-map, ncomb-map and rms-map.
ZPL_ MODEM_ EFF_ QPLUS	FITS[Im(8)]	The resulting +Q modem efficiency frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions: modem efficiency qplus intensity image, badpixel-map, ncomb-map and rms-map. modem efficiency qplus polarization image, badpixel-map, ncomb-map and rms-map.
ZPL_ MODEM_ EFF_ QPLUS_ CAM1	FITS[Im(8)]	The resulting +Q modem efficiency frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions: modem efficiency qplus intensity image, badpixel-map, ncomb-map and rms-map. modem efficiency qplus polarization image, badpixel-map, ncomb-map and rms-map.

Name	Type	Description
ZPL_ MODEM_ EFF_ QMINUS	FITS[Im(8)]	The resulting -Q modem efficiency frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions: modem efficiency qminus intensity image, badpixel-map, ncomb-map and rms-map. modem efficiency qminus polarization image, badpixel-map, ncomb-map and rms-map.
ZPL_ MODEM_ EFF_ QMINUS_ CAM1	FITS[Im(8)]	The resulting -Q modem efficiency frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions: modem efficiency qminus intensity image, badpixel-map, ncomb-map and rms-map. modem efficiency qminus polarization image, badpixel-map, ncomb-map and rms-map.
ZPL_ MODEM_ EFF_ CAM2	FITS[Im(4)]	The final modem efficiency frame is of the MASTER FRAME format. This frame contains 4 image extensions: modem efficiency image, badpixel-map, ncomb-map and rms-map.
ZPL_ MODEM_ EFF_ QPLUS_ CAM2	FITS[Im(8)]	The resulting +Q modem efficiency frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions: modem efficiency qplus intensity image, badpixel-map, ncomb-map and rms-map. modem efficiency qplus polarization image, badpixel-map, ncomb-map and rms-map.
ZPL_ MODEM_ EFF_ QMINUS_ CAM2	FITS[Im(8)]	The resulting -Q modem efficiency frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions: modem efficiency qminus intensity image, badpixel-map, ncomb-map and rms-map. modem efficiency qminus polarization image, badpixel-map, ncomb-map and rms-map.

10.11 sph_zpl_aoc_efficiency

Purpose:

Measure AOC efficiency, polarimetric modes (not currently applicable).

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_AOC_EFF_PREPROC_RAW	Raw data	No	1	Any
ZPL_MASTER_BIAS	Calibration	Yes	0	1
ZPL_MASTER_DARK	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD_MASTER	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD	Calibration	Yes	0	1
ZPL_POL_FLAT_FIELD	Calibration	Yes	0	1
ZPL_MODEM_EFF	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	>>SPH PC PREPROC ZPL EXP<<
ESO DRS ZPL HWP2 SETTING	double	No	SPH_ZPL_KEYWORD_STOCK_PARAMETER_SIGN string 0 0 0 >>PLUS<< >>MINUS<< SPH_ZPL_KEYWORD_DEROTATOR_ANGLE float 0 0 360.0 The rotation angle of frame in degrees.

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.aoc_efficiency.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	zpl_aoc_efficiency.fits	-
zpl.aoc_efficiency_plus.outfilename	string	The output filename for the plus product. Please also see the esorex documentation for naming of output products.	zpl_aoc_efficiency_plus.fits	-
zpl.aoc_efficiency_minus.outfilename	string	The output filename for the minus product. Please also see the esorex documentation for naming of output products.	zpl_aoc_efficiency_minus.fits	-
zpl.aoc_efficiency.subtract_overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.aoc_efficiency.coll_alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median, 2 = Clean Mean. Default is 2 = Clean Mean	2	0,1,2
zpl.aoc_efficiency.coll_alg.clean_mean.reject_high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low	0	0-20

Name	Type	Description	Default	Allowed vals.
zpl.aoc_ efficiency.coll_ alg.clean_ mean.reject_ low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low	0	0-20

Description:

The recipe creates master AOC efficiency calibration frame, using the input exposures which should be taken as described in the ZPL calibration plan. The input raw frames used by this recipe are the pre-processed raw aoc efficiency frames of two groups for several de-rotator angle position [0, 2PI], distinguished from each other by the opposite sign of the stock parameter [PLUS, MINUS] These raw frames should carry the ZPL_AOC_EFF_PREPROC_RAW tag (normally, it must be raw frames with one particular DIT and filter).The input calibration frames (if any) are the master bias frame with BIAS_CALIB tag, master dark frame with the DARK_CALIB tag, the master intensity flat field frame with the IFF_CALIB tag, the polarization flat field with the PFF_CALIB tag, andmodulation / demodulation efficiency with the MODEM_CALIB tag.After reading the pre-processed aoc polarization efficiency raw frames, they are grouped by two with regard to the sign of the stock parameter [PLUS, MINUS]for each de-rotator position. The frames from each group are combined using the specified collapse algorithm (usually the clean_mean algorithm, defined as a default one) for each zpl exposure sub-frames.The combined frames are of the QUAD IMAGE (16 extensions) format specified as follows:- zpl exp phase zero odd sub-frame combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase zero even sub-frame combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase PI odd sub-frame master combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase PI even sub-frame master combined image, badpixel-map, ncomb-map and rms-map;The master bias, dark and intensity flat field are applied to the two [PLUS, MINUS] combined master frames and then the stock parameters (I,P) are calculated for both group of frames for each de-rotator position in the form of DOUBLE IMAGE:- master intensity stock parameter image, badpixel-map, ncomb-map and rms-map;- master polarization stock parameter image, badpixel-map, ncomb-map and rms-map.Then, the polarization flat field and modem efficiency are applied to the [PLUS, MINUS] stock parameters double image frames.and the two opposite polarization frames are combined for each de-rotator position by subtracting the MINUS polarization image frame from the PLUS one.The output master aoc polarization efficiency frame for one given de-rotator position is of the MASTER FRAME format specified as follows:aoc efficiency image, badpixel-map, rms-map and weight-map. Finally, recipe performs a fit for the mean values (SPH_COMMON_KEYWORD_QC_MEANMASTERFRAME) with it standard deviation (SPH_COMMON_KEYWORD_QC_RMSMASTERFRAME) of the obtained master frameswith respect to its de-rotator positions. The fitting coefficients are written out in the header of the productfits file. This master AOC efficiency product is used in the science polarization recipes and in all on sky polarization calibration recipes.

Products:

Name	Type	Description
ZPL_ AOC_ EFF	FITS[Im(4)]	The resulting Q->Q, U->U or V->V master aoc polarization efficiency frame is of the MASTER FRAME format. This frame contains 4 image extensions: - master aoc efficiency image, badpixel-map, ncomb-map and rms-map.

Name	Type	Description
ZPL_ AOC_ EFF_ PLUS	FITS[Im(8)]	The resulting +Q/+U aoc polarization efficiency frame is of the DOUBLE IMAGE format. This frame contains 4 image extensions: - aoc efficiency qplus intensity image, badpixel-map, ncomb-map and rms-map. - aoc efficiency qplus polarization image, badpixel-map, ncomb-map and rms-map.
ZPL_ AOC_ EFF_ MINUS	FITS[Im(8)]	The resulting -Q/-U aoc polarization efficiency frame is of the DOUBLE IMAGE format. This frame contains 4 image extensions: - aoc efficiency qminus intensity image, badpixel-map, ncomb-map and rms-map. - aoc efficiency qminus polarization image, badpixel-map, ncomb-map and rms-map.

10.12 sph_zpl_aoc_offset

Purpose:

Measure AOC polarization offset, polarimetric modes (not currently applicable).

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ AOC_ OFFSET_ PREPROC_ RAW	Raw data	No	1	Any
ZPL_ MASTER_ BIAS	Calibration	Yes	0	1
ZPL_ MASTER_ DARK	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ MASTER	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD	Calibration	Yes	0	1
ZPL_ POL_ FLAT_ FIELD	Calibration	Yes	0	1
ZPL_ MODEM_ EFF	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	>>SPH PC PREPROC ZPL EXP<<
ESO DRS ZPL HWP2 SETTING	double	No	SPH_ZPL_KEYWORD_STOCK_PARAMETER_SIGN string 0 0 0 SPH_ZPL_KEYWORD_STOCK_PARAMETER_SIGN must be substituted by a keyword associated with HWP2 >>PLUS<< >>MINUS<< SPH_ZPL_KEYWORD_DEROTATOR_ANGLE float 0 0 360.0 The rotation angle of frame in degrees.

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.aoc_offset.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	zpl_aoc_offset.fits	-
zpl.aoc_offset_plus.outfilename	string	The output filename for the plus product. Please also see the esorex documentation for naming of output products.	zpl_aoc_offset_plus.fits	-
zpl.aoc_offset_minus.outfilename	string	The output filename for the minus product. Please also see the esorex documentation for naming of output products.	zpl_aoc_offset_minus.fits	-
zpl.aoc_offset.subtract_overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.aoc_offset.coll_alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median, 2 = Clean Mean. Default is 2 = Clean Mean	2	0,1,2
zpl.aoc_offset.coll_alg.clean_mean.reject_high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.aoc_offset.coll_alg.clean_mean.reject_low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low	0	0-20

Description:

The recipe creates master AOC polarization offset calibration frame, using the input exposures which should be taken as described in the ZPL calibration plan. The input raw frames used by this recipe are the pre-processed raw aoc offset frames of two groups for several de-rotator angle position [0, 2PI], distinguished from each other by the opposite sign of the stockparameter [PLUS, MINUS]. These raw frames should carry the ZPL_AOC_OFFSET_PREPROC_RAW tag (normally, it must be raw frames with one particular DIT and filter).The input calibration frames (if any) are the master bias frame with BIAS_CALIB tag, master dark frame with the DARK_CALIB tag, the master intensity flat field frame with the IFF_CALIB tag, the polarization flat field with the PFF_CALIB tag,modulation / demodulation efficiency with the MODEM_CALIB tag, and AOC polarization efficiency with the AOCEFF_CALIB tag.After reading the pre-processed aoc polarization offset raw frames are grouped by two with regard to the sign of the stock parameter [PLUS, MINUS] for each de-rotator position. The frames from each group are combined using the specified collapse algorithm (usually the clean_mean algorithm, defined as a default one) for each zpl exposure sub-frames.The combined frame is of the QUAD IMAGE (16 extensions) format specified as follows:- zpl exp phase zero odd sub-frame combined image, badpixel-map, rms-map and weight-map;- zpl exp phase zero even sub-frame combined image, badpixel-map, rms-map and weight-map;- zpl exp phase PI odd sub-frame master combined image, badpixel-map, rms-map and weight-map;- zpl exp phase PI even sub-frame master combined image, badpixel-map, rms-map and weight-map.The master bias, dark and intensity flat field are applied to the two [PLUS, MINUS] combined master frames and then the stock parameters (I,P) are calculated for both group of frames for each de-rotator position in the form of DOUBLE IMAGE:- master intensity stock parameter image, badpixel-map, ncomb-map and rms-map;- master polarization stock parameter image, badpixel-map, ncomb-map and rms-map.Then, the polarization flat field and modem efficiency are applied to the [PLUS, MINUS] stock parameters double image frames.Then, the two opposite polarization frames are averaged by polarization images. The output master aoc polarization offset frame for one given de-rotator position is of the MASTER FRAME format specified as follows: aoc offset image, badpixel-map, rms-map and weight-map.Finally, recipe performs a fit for the mean values (SPH_COMMON_KEYWORD_QC_MEANMASTERFRAME) with it standard deviation (SPH_COMMON_KEYWORD_QC_RMSMASTERFRAME) of the obtained master frameswith respect to its de-rotator positions. The fitting coefficients are written out in the header of the productfits file. This master AOC polarization offset product is used in the science polarization recipes.

Products:

Name	Type	Description
ZPL_ AOC_ OFFSET	FITS[Im(4)]	The resulting I->Q, I->U aoc polarization offset frame is of the MASTER FRAME format. This frame contains 4 image extensions: - master aoc offset image, badpixel-map, ncomb-map and rms-map.
ZPL_ AOC_ OFFSET_ PLUS	FITS[Im(8)]	The resulting +Q/+U aoc polarization efficiency frame is of the DOUBLE IMAGE format. This frame contains 4 image extensions: - aoc offset plus intensity image, badpixel-map, ncomb-map and rms-map. - aoc offset plus polarization image, badpixel-map, ncomb-map and rms-map.

Name	Type	Description
ZPL_AOC_OFFSET_MINUS	FITS[Im(8)]	The resulting -Q/-U aoc polarization efficiency frame is of the DOUBLE IMAGE format. This frame contains 4 image extensions: - aoc offset minus intensity image, badpixel-map, ncomb-map and rms-map. - aoc offset minus polarization image, badpixel-map, ncomb-map and rms-map.

10.13 sph_zpl_aoc_crosstalk

Purpose:

Measure AOC crosstalk, polarimetric modes (not currently applicable).

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_AOC_CROSSTALK_PREPROC_RAW	Raw data	No	1	Any
ZPL_MASTER_BIAS	Calibration	Yes	0	1
ZPL_MASTER_DARK	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD_MASTER	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD	Calibration	Yes	0	1
ZPL_POL_FLAT_FIELD	Calibration	Yes	0	1
ZPL_MODEM_EFF	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	>>SPH PC PREPROC ZPL EXP<<
ESO DRS ZPL HWP2 SETTING	double	No	SPH_ZPL_KEYWORD_STOCK_PARAMETER_SIGN string 0 0 0 >>PLUS<< >>MINUS<< SPH_ZPL_KEYWORD_DEROTATOR_ANGLE float 0 0 360.0 The rotation angle of frame in degrees.

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.aoc_crosstalk.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	zpl_aoc_crosstalk.fits	-

Name	Type	Description	Default	Allowed vals.
zpl.aoc_ crosstalk_ plus.outfilename	string	The output filename for the plus product. Please also see the esorex documentation for naming of output products.	zpl_ aoc_ crosstalk_ plus.fits	-
zpl.aoc_ crosstalk_ minus.outfilename	string	The output filename for the minus product. Please also see the esorex documentation for naming of output products.	zpl_ aoc_ crosstalk_ minus.fits	-
zpl.aoc_ crosstalk.subtract_ overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.aoc_ crosstalk.coll_ alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median, 2 = Clean Mean. Default is 2 = Clean Mean	2	0,1,2
zpl.aoc_ crosstalk.coll_ alg.clean_ mean.reject_ high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.aoc_ crosstalk.coll_ alg.clean_ mean.reject_ low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low	0	0-20

Description:

The recipe creates master AOC crosstalk calibration frame, using the input exposures which should be taken as described in the ZPL calibration plan. The input raw frames used by this recipe are the pre-processed raw aoc crosstalk frames of two groups for several de-rotator angle position [0, 2PI], distinguished from each other by the opposite sign of the stock parameter [PLUS, MINUS]. These raw frames should carry the ZPL_AOC_CROSSTALK_PREPROC_RAW tag (normally, it must be raw frames with one particular DIT and filter).The input calibration frames (if any) are the master bias frame with BIAS_CALIB tag, master dark frame with the DARK_CALIB tag, the master intensity flat field frame with the IFF_CALIB tag, the polarization flat field with the PFF_CALIB tag, andmodulation / demodulation efficiency with the MODEM_CALIB tag.After reading the pre-processed aoc crosstalk raw frames, they are grouped by two with regard to the sign of the stock parameter [PLUS, MINUS] for each de-rotator position.The frames from each group are then combined using the specified collapse algorithm (usually the clean_mean algorithm, defined as a default one) for each zpl exposure sub-frames.The combined frame is of the QUAD IMAGE (16 extensions) format specified as follows:- zpl exp phase zero odd sub-frame combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase zero even sub-frame combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase PI odd sub-frame master combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase PI even sub-frame master combined image, badpixel-map, rms-map and rms-map.The master bias, dark and intensity flat field are applied to the two [PLUS, MINUS] combined master frames and then the stock parameters (I,P) are calculated for both frames in the form of DOUBLE IMAGE:- master intensity stock parameter image, badpixel-map, ncomb-map and rms-map;- master polarization stock parameter image, badpixel-map, ncomb-map and rms-map.Then, the polarization flat field and modem efficiency are applied to the [PLUS, MINUS] stock parameters double image frames.Finally, the two opposite polarization frames are combined for each de-rotator position by subtracting the MINUS polarization image frame from the PLUS one. The output master polarization crosstalk frame for one given de-rotator position is of the MASTER FRAME format specified as follows:aoc crosstalk image, badpixel-map, rms-map and weight-map.Finally, recipe performs a fit for the mean values (SPH_COMMON_KEYWORD_QC_MEANMASTERFRAME) with it standard deviation (SPH_COMMON_KEYWORD_QC_RMSMASTERFRAME) of the obtained master frameswith respect to its de-rotator positions. The fitting coefficients are written out in the header of the productfits file. This master AOC crosstalk product is used in the science polarization recipes and in all >>on sky<< polarization calibration recipes.

Products:

Name	Type	Description
ZPL_ AOC_ CROSSTALK	FITS[Im(4)]	The resulting master aoc polarization crosstalk frame is of the MASTER FRAME format. This frame contains 4 image extensions: master aoc crosstalk image, badpixel-map, ncomb-map and rms-map.
ZPL_ AOC_ OFFSET_ PLUS	FITS[Im(8)]	The resulting aoc polarization crosstalk frame is of the DOUBLE IMAGE format. This frame contains 4 image extensions: - aoc crosstalk plus intensity image, badpixel-map, ncomb-map and rms-map. - aoc crosstalk plus polarization image, badpixel-map, ncomb-map and rms-map.
ZPL_ AOC_ OFFSET_ MINUS	FITS[Im(8)]	The resulting aoc polarization efficiency frame is of the DOUBLE IMAGE format. This frame contains 4 image extensions: - aoc crosstalk minus intensity image, badpixel-map, ncomb-map and rms-map. - aoc crosstalk minus polarization image, badpixel-map, ncomb-map and rms-map.

10.14 sph_zpl_zimpol_crosstalk

Purpose:

Measure zimpol crosstalk, polarimetric modes (not currently applicable).

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ZIMPOL_CROSSTALK_PREPROC_RAW	Raw data	No	1	Any
ZPL_MASTER_BIAS	Calibration	Yes	0	1
ZPL_MASTER_DARK	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD_MASTER	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD	Calibration	Yes	0	1
ZPL_POL_FLAT_FIELD	Calibration	Yes	0	1
ZPL_MODEM_EFF	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	>>SPH PC PREPROC ZPL EXP<< SPH_ZPL_KEYWORD_STOCK_PARAMETER_SIGN string 0 0 0 >>PLUS<< >>MINUS<<

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.zimpol_crosstalk.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	zpl_zimpol_crosstalk.fits	-
zpl.zimpol_crosstalk_plus.outfilename	string	The output filename for the plus product. Please also see the esorex documentation for naming of output products.	zpl_zimpol_crosstalk_plus.fits	-
zpl.zimpol_crosstalk_minus.outfilename	string	The output filename for the minus product. Please also see the esorex documentation for naming of output products.	zpl_zimpol_crosstalk_minus.fits	-
zpl.zimpol_crosstalk.subtract_overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.zimpol_crosstalk.coll_alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median, 2 = Clean Mean. Default is 2 = Clean Mean	2	0,1,2
zpl.zimpol_crosstalk.coll_alg.clean_mean.reject_high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.zimpol_crosstalk.coll_alg.clean_mean.reject_low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low	0	0-20

Description:

The recipe creates master zimpol crosstalk calibration frame, using the input exposures which should be taken as described in the ZPL calibration plan. The input raw frames used by this recipe are the pre-processed raw zimpol crosstalk frames of two groups, distinguished from each other by the opposite sign of the stock parameter [PLUS, MINUS]. These frames should carry the ZPL_ZIMPOL_CROSSTALK_PREPROC_RAW tag (normally, it must be raw frames with one particular DIT and filter),The input calibration frames (if any) are the master bias frame with BIAS_CALIB tag, master dark frame with the DARK_CALIB tag, the master intensity flat field frame with the IFF_CALIB tag, and the polarization flat field with the PFF_CALIB tag.The pre-processed modem raw frames are grouped by two with regard to the sign of the stock parameter [PLUS, MINUS].The frames from each group are combined using the specified collapse algorithm (usually the clean_mean algorithm, defined as a default one) for each zpl exposure sub-frames.The combined frame is of the QUAD IMAGE (16 extensions) format specified as follows:- zpl exp phase zero odd sub-frame combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase zero even sub-frame combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase PI odd sub-frame master combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase PI even sub-frame master combined image, badpixel-map, ncomb-map and rms-map.The master bias, dark and intensity flat field are applied to the two [PLUS, MINUS] combined master frames and then the stock parameters (I,P) are calculated for both frames in the form of DOUBLE IMAGE:- master intensity stock parameter image, badpixel-map, ncomb-map and rms-map;- master polarization stock parameter image, badpixel-map, ncomb-map and rms-map.Then, the polarization flat field and modem efficiency calibrations are applied to the [PLUS, MINUS] stock parameters double image frames.Finally, the two opposite polarization frames are combined by subtracting the MINUS polarization image frame from the PLUS one. The output master zimpol polarization crosstalk is of the MASTER FRAME format specified as follows:zimpol crosstalk image, badpixel-map, ncomb-map and rms-map.This master zimpol crosstalk product is used in polarization science recipes.

Products:

Name	Type	Description
ZPL_ ZIMPOL_ CROSSTALK	FITS[Im(4)]	The resulting master zimpol crosstalk frame is of the MASTER FRAME format. This frame contains 4 image extensions: master zimpol crosstalk image, badpixel-map, rms-map and weight-map.
ZPL_ ZIMPOL_ CROSSTALK_ PLUS	FITS[Im(8)]	The resulting +U/+V -> Q zimpol crosstalk frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - zimpol crosstalk plus intensity image, badpixel-map, rms-map and weight-map. - zimpol crosstalk plus polarization image, badpixel-map, rms-map and weight-map.

Name	Type	Description
ZPL_ ZIMPOL_ CROSSTALK_ MINUS	FITS[Im(8)]	The resulting -U/-V -> Q zimpol crosstalk frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - zimpol crosstalk minus intensity image, badpixel-map, rms-map and weight-map. - zimpol crosstalk minus polarization image, badpixel-map, rms-map and weight-map.

10.15 sph_zpl_basic_polarization

Purpose:

Calibrate frames, polarimetric modes (utility recipe).

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ BASIC_ POL_ PREPROC_ RAW	Raw data	No	1	Any
ZPL_ MASTER_ BIAS	Calibration	Yes	0	1
ZPL_ MASTER_ DARK	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD	Calibration	Yes	0	1
ZPL_ POL_ FLAT_ FIELD	Calibration	Yes	0	1
ZPL_ MODEM_ EFF	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if the pre-processed data are used. As the format of the zimpol pre-processed data is complicated, this keyword was introduced in order to garantee that the pre-processed input frames are polarimetric pre-processed data, produced by the sph_zpl_preproc utility recipe.

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.basic_ polarization.outfilename	string	The prefix for the output filename for the product. Please also see the esorex documentation for naming of output products.	zpl_ basic_ polarization.fits	-

Name	Type	Description	Default	Allowed vals.
zpl.basic_polarization.subtract_overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-

Description:

The recipe produces calibrated frames in the polarization modes.This recipe is an utility recipe and should be only used by off-line data reduction! The input raw frames used by this recipe are the already pre-processed polarimetric data. These frames should carry the ZPL_BASIC_POL_PREPROC_RAW tag (normally, it must be raw frames with one particular DIT and filter),The input calibration frames (if any) are the master bias frame with BIAS_CALIB tag, master dark frame with the DARK_CALIB tag, the master intensity flat field frame with the IFF_CALIB tag, the polarization flat field with the PFF_CALIB tag, and the modem efficiencyframe with the MODEM_EFF_CALIB tag.The pre-processed raw frames are transformed first to the QUAD IMAGE (16 extensions) format specified as follows:- zpl exp phase zero odd sub-frame combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase zero even sub-frame combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase PI odd sub-frame master combined image, badpixel-map, rms-map and rms-map;- zpl exp phase PI even sub-frame master combined image, badpixel-map, rms-map and rms-map.The master bias, dark and intensity flat field are applied to the quad images, then the stock parameters (I,P) are calculated from each quad image, creation double images with the DOUBLE IMAGE format:- master intensity stock parameter image, badpixel-map, ncomb-map and weight-map;- master polarization stock parameter image, badpixel-map, ncomb-map and weight-map.Then, the polarization flat field is applied to the stock parameters double image frames.Finally, the modem efficiency is applied to the stock parameters double image frames. The output basic polarimetry calibrated frames are saved as double image frames.

Products:

Name	Type	Description
ZPL_ BASIC_ POL_ CALIBRATED	FITS[Im(8)]	The resulting basic calibration frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - basic polarimetric calibrated intensity image, badpixel-map, ncomb-map and rms-map. - basic polarimetric calibrated polarimetry image, badpixel-map, ncomb-map and rms-map.

10.16 sph_zpl_basic_imaging

Purpose:

Calibrate frames, imaging mode (utility recipe).

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ BASIC_ IMAGING_ PREPROC_ RAW	Raw data	No	1	Any
ZPL_ MASTER_ BIAS_ IMAGING	Calibration	Yes	0	1
ZPL_ MASTER_ DARK_ IMAGING	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ IMAGING	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if the pre-processed data are used. As the format of the zimpol pre-processed data is complicated, this keyword was introduced in order to garantee that the pre-processed input frames are polarimetric pre-processed data, produced by the sph_zpl_preproc_imaging utility recipe.

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.basic_ imaging.outfilename	string	The prefix for the output filename for the product. Please also see the esorex documentation for naming of output products.	zpl_ basic_ imaging.fits	-
zpl.basic_ imaging.subtract_ overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-

Description:

The recipe produces calibrated frames for the imaging mode. The input raw frames used by this recipe are already pre-processed imaging data. These frames should carry the ZPL_BASIC_IMAGING_PREPROC_RAW tag (normally, it must be raw frames with one particular DIT and filter). The input calibration frames (if any) are the master bias frame with BIAS_IMAGING_CALIB tag, master dark frame with the DARK_IMAGING_CALIB tag, the intensity flat field frame with the IFF_IMAGING CALIB tag.The pre-processed raw frames are transformed first into the DOUBLE IMAGE (8 extensions) format specified as follows:- odd subframe image (intensity), badpixel-map, ncomb-map and rms-map;- even sub-frame image (dark current/non informative), badpixel-map, ncomb-map and rms-map.The master bias, dark and intensity flat field are applied to the double images.The output basic imaging calibrated frames are saved as double image frames.

Products:

Name	Type	Description
ZPL_ BASIC_ IMAGING_ CALIBRATED	FITS[Im(8)]	The resulting basic calibration frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - basic imaging calibrated intensity image, badpixel-map, ncomb-map and rms-map. - basic imaging calibrated dark image (non-informative), badpixel-map, ncomb-map and rms-map.

10.17 sph_zpl_basic_collapse_polarization

Purpose:

Calibrate and collapse frames, polarimetric modes (utility recipe).

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ BASIC_ COLL_ POL_ PREPROC_ RAW	Raw data	No	1	Any
ZPL_ MASTER_ BIAS	Calibration	Yes	0	1
ZPL_ MASTER_ DARK	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD	Calibration	Yes	0	1
ZPL_ POL_ FLAT_ FIELD	Calibration	Yes	0	1
ZPL_ MODEM_ EFF	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if the pre-processed data are used. As the format of the zimpol pre-processed data is complicated, this keyword was introduced in order to garantee that the pre-processed input frames are polarimetric pre-processed data, produced by the sph_zpl_preproc utility recipe.

Parameters:

Name	Type	Description	Default	Allowed vals.

Name	Type	Description	Default	Allowed vals.
zpl.basic_collapse_polarization.outfilename	string	The prefix for the output filename for the product. Please also see the esorex documentation for naming of output products.	zpl_basic_collapse_polarization.fits	-
zpl.basic_collapse_polarization_plus.outfilename	string	The prefix for the output filename for the +Q product. Please also see the esorex documentation for naming of output products.	zpl_basic_collapse_polarization_plus.fits	-
zpl.basic_collapse_polarization_minus.outfilename	string	The prefix for the output filename for the -Q product. Please also see the esorex documentation for naming of output products.	zpl_basic_collapse_polarization_minus.fits	-
zpl.basic_collapse_polarization.subtract_overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.basic_collapse_polarization.coll_alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median, 2 = Clean Mean. Default is 2 = Clean Mean	2	0,1,2
zpl.basic_collapse_polarization.coll_alg.clean_mean.reject_high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.basic_collapse_polarization.coll_alg.clean_mean.reject_low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.basic_collapse_polarization.clean_mean.sigma	double	The number of pixels to reject when combining frames in sigma from median. NOT SUPPORTED YET!	5.0	0.0-200.0
zpl.basic_collapse_polarization.sigma_clip	double	The sigma clipping value for static badpixel detection. Default is 5.	5.0	0.0-200.0
zpl.basic_collapse_polarization.cfg_filename	string	The configuration filename containing information about the planes which should be ignored at collapsing.	zpl_basic_collapse_polarization.cfg	-

Description:

The recipe produces calibrated and collapsed frame in the polarization modes (simple science recipe – no de-dithering and de-rotation). This recips is an utility recipe and should be only used by off-line data reduction! The input raw frames used by this recipe are already pre-processed polarimetric data. These frames should carry the ZPL_BASIC_COLL_POL_PREPROC_RAW tag (normally, it must be raw frames with one particular DIT and filter),The input calibration frames (if any) are the master bias frame with BIAS_CALIB tag, master dark frame with the DARK_CALIB tag, the master intensity flat field frame with the IFF_CALIB tag, the polarization flat field with the PFF_CALIB tag, and the modem efficiencyframe with the MODEM_EFF_CALIB tag.The pre-processed raw frames are collapse first, producing the QUAD IMAGE (16 extensions) frame specified as follows:- zpl exp phase zero odd sub-frame combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase zero even sub-frame combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase PI odd sub-frame master combined image, badpixel-map, ncomb-map and rms-map;- zpl exp phase PI even sub-frame master combined image, badpixel-map, ncomb-map and rms-map.The master bias, dark and intensity flat field are applied to the quad images, then the stock parameters (I,P) are calculated from each quad image, creation double images with the DOUBLE IMAGE format:- master intensity stock parameter image, badpixel-map, ncomb-map and rms-map;- master polarization stock parameter image, badpixel-map, ncomb-map and rms-map.Then, the polarization flat field is applied to the stock parameters double image frames.Finally, the modem efficiency is applied to the stock parameters double image frames. The output basic polarimetry collapsed product is saved as double image frame.

Products:

Name	Type	Description
ZPL_ BASIC_ COLL_ POL_ REDUCED	FITS[Im(8)]	The resulting basic collapsed reduced frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - basic polarimetric calibrated intensity image, badpixel-map, rms-map and weight-map. - basic polarimetric calibrated polarimetry image, badpixel-map, rms-map and weight-map.
ZPL_ BASIC_ COLL_ POL_ PLUS_ REDUCED	FITS[Im(8)]	The resulting +Q basic collapsed reduced frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - basic polarimetric collapsed qplus intensity image, badpixel-map, rms-map and weight-map. - basic polarimetric collapsed qplus polarization image, badpixel-map, rms-map and weight-map.
ZPL_ BASIC_ COLL_ POL_ MINUS_ REDUCED	FITS[Im(8)]	The resulting -Q basic collapsed reduced frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - basic collapsed qminus intensity image, badpixel-map, rms-map and weight-map. - basic collapsed qminus polarization image, badpixel-map, rms-map and weight-map.

10.18 sph_zpl_basic_collapse_imaging

Purpose:

Calibrate and collapse frames, imaging mode (utility recipe).

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ BASIC_ COLLAPSE_ IMAGING_ PREPROC_ RAW	Raw data	No	1	Any
ZPL_ MASTER_ BIAS_ IMAGING	Calibration	Yes	0	1
ZPL_ MASTER_ DARK_ IMAGING	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ IMAGING	Calibration	Yes	0	1

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if the pre-processed data are used. As the format of the zimpol pre-processed data is complicated, this keyword was introduced in order to garantee that the pre-processed input frames are polarimetric pre-processed data, produced by the sph_zpl_preproc_imaging utility recipe.

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.basic_ collapse_ imaging.outfilename	string	The prefix for the output filename for the product. Please also see the esorex documentation for naming of output products.	zpl_ basic_ collapse_ imaging.fits	-
zpl.basic_ collapse_ imaging.subtract_ overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.basic_ collapse_ imaging.coll_ alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median, 2 = Clean Mean. Default is 2 = Clean Mean	2	0,1,2
zpl.basic_ collapse_ imaging.coll_ alg.clean_ mean.reject_ high	int	The number of pixels to reject when combining frames at the high end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.basic_ collapse_ imaging.coll_ alg.clean_ mean.reject_ low	int	The number of pixels to reject when combining frames at the low end. Number of input frames must be > reject_high +reject_low	0	0-20
zpl.basic_ collapse_ imaging.clean_ mean.sigma	double	The number of pixels to reject when combining frames in sigma from median. NOT SUPPORTED YET!	5.0	0.0-200.0

Name	Type	Description	Default	Allowed vals.
zpl.basic_ collapse_ imaging.sigma_ clip	double	The sigma clipping value for static badpixel detection. Default is 5.	5.0	0.0-200.0
zpl.basic_ collapse_ imaging.cfg_ filename	string	The configuration filename containing information about the planes which should be ignored at collapsing.	zpl_ basic_ collapse_ imaging.cfg	-

Description:

The recipe produces calibrated and collpased frame in the imaging mode (simple science recipe – no de-dithering and de-rotation).This recips is an utility recipe and should be only used by off-line data reduction! The input raw frames used by this recipe are already pre-processed imaging data. These frames should carry the ZPL_BASIC_IMAGING_PREPROC_RAW tag (normally, it must be raw frames with one particular DIT and filter). The input calibration frames (if any) are the master bias frame with BIAS_IMAGING_CALIB tag, master dark frame with the DARK_IMAGING_CALIB tag, the intensity flat field frame with the IFF_IMAGING CALIB tag.The pre-processed raw frames are transformed first into the DOUBLE IMAGE (8 extensions) format specified as follows:- odd sub-frame image (intensity), badpixel-map, ncomb-map and rms-map;- even sub-frame image (dark current/non informative), badpixel-map, ncomb-map and rms-map.The master bias, dark and intensity flat field are applied to the double images.The output basic imaging calibrated frames are saved as double image frame.

Products:

Name	Type	Description
ZPL_ BASIC_ COLLAPSE_ IMAGING_ REDUCED	FITS[Im(8)]	The resulting basic calibration frame is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - basic imaging calibrated intensity image, badpixel-map, rms-map and weight-map. - basic imaging calibrated dark image (non-informative), badpixel-map, rms-map and weight-map.

10.19 sph_zpl_science_p1

Purpose:

Reduce science frames of the Q and/or U observations in the polarization P1 mode.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_SCIENCE_P1_RAW	Raw data	Yes	0	Any
ZPL_SCIENCE_P1_PREPROC	Calibration	Yes	0	Any
ZPL_SCIENCE_P1_PREPROC_CAM1	Calibration	Yes	0	Any
ZPL_SCIENCE_P1_PREPROC_CAM2	Calibration	Yes	0	Any
ZPL_MASTER_BIAS	Calibration	Yes	0	1
ZPL_MASTER_BIAS_CAM1	Calibration	Yes	0	1
ZPL_MASTER_BIAS_CAM2	Calibration	Yes	0	1
ZPL_MASTER_DARK	Calibration	Yes	0	1
ZPL_MASTER_DARK_CAM1	Calibration	Yes	0	1
ZPL_MASTER_DARK_CAM2	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD_CAM1	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD_CAM2	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD_MASTER	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD_MASTER_CAM1	Calibration	Yes	0	1
ZPL_INT_FLAT_FIELD_MASTER_CAM2	Calibration	Yes	0	1
ZPL_POL_FLAT_FIELD	Calibration	Yes	0	1
ZPL_POL_FLAT_FIELD_CAM1	Calibration	Yes	0	1
ZPL_POL_FLAT_FIELD_CAM2	Calibration	Yes	0	1
ZPL_MODEM_EFF	Calibration	Yes	0	1
ZPL_MODEM_EFF_CAM1	Calibration	Yes	0	1
ZPL_MODEM_EFF_CAM2	Calibration	Yes	0	1
ZPL_CENTER_TABLE	Calibration	Yes	0	Any

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if the pre-processed data are used. As the format of the zimpol pre-processed data is complicated, this keyword was introduced in order to garantee that the pre-processed input frames are polarimetric pre-processed data, produced by the sph_zpl_preproc utility recipe. Note: if raw data are used (default), then all keywords needed for the pre-processing recipe (see sph_zpl_preproc) must be presented in the raw data.
ESO OCS3 ZIMPOL POL STOKES	string	No	Stokes parameters (Qplus, Qminus, Uplus, Uminus) SPH_COMMON_KEYWORD_CAM1_DITHERING_X double 0 0 100.0 X-position of the arm1(camera-1) [pix] SPH_COMMON_KEYWORD_CAM1_DITHERING_Y double 0 0 100.0 Y-position of the arm1(camera-1) [pix] SPH_COMMON_KEYWORD_CAM2_DITHERING_X double 0 0 100.0 X-position of the arm2(camera-2) [pix] SPH_COMMON_KEYWORD_CAM2_DITHERING_Y double 0 0 100.0 Y-position of the arm2(camera-2) [pix] SPH_COMMON_KEYWORD_DROT2_MODE string 0 0 0 De-rotator mode: ELEV(pupil stabilized), SKY(field stabilized)

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.science_ p1.outfilename_ q	string	The output filename for the final science product Q. Please also see the esorex documentation for naming of output products.	zpl_ science_ p1_ q.fits	-
zpl.science_ p1_ plus_ q.outfilename	string	The output filename for the science plus product +Q. Please also see the esorex documentation for naming of output products.	zpl_ science_ p1_ plus_ q.fits	-
zpl.science_ p1_ minus_ q.outfilename	string	The output filename for the science minus product -Q. Please also see the esorex documentation for naming of output products.	zpl_ science_ p1_ minus_ q.fits	-
zpl.science_ p1.outfilename_ u	string	The output filename for the final science product U. Please also see the esorex documentation for naming of output products.	zpl_ science_ p1_ u.fits	-
zpl.science_ p1_ plus_ u.outfilename	string	The output filename for the science plus product +U. Please also see the esorex documentation for naming of output products.	zpl_ science_ p1_ plus_ u_ cam1.fits	-

Name	Type	Description	Default	Allowed vals.
zpl.science_p1_minus_u.outfilename	string	The output filename for the science minus product -U. Please also see the esorex documentation for naming of output products.	zpl_science_p1_minus_u.fits	-
zpl.science_p1.outfilename_q_cam1	string	The output filename for the final science product Q. Please also see the esorex documentation for naming of output products.	zpl_science_p1_q_cam1.fits	-
zpl.science_p1_plus_q_cam1.outfilename	string	The output filename for the science plus product +Q. Please also see the esorex documentation for naming of output products.	zpl_science_p1_plus_q_cam1.fits	-
zpl.science_p1_minus_q_cam1.outfilename	string	The output filename for the science minus product -Q. Please also see the esorex documentation for naming of output products.	zpl_science_p1_minus_q_cam1.fits	-
zpl.science_p1.outfilename_u_cam1	string	The output filename for the final science product U. Please also see the esorex documentation for naming of output products.	zpl_science_p1_u_cam1.fits	-
zpl.science_p1_plus_u_cam1.outfilename	string	The output filename for the science plus product +U. Please also see the esorex documentation for naming of output products.	zpl_science_p1_plus_u_cam1.fits	-
zpl.science_p1_minus_u_cam1.outfilename	string	The output filename for the science minus product -U. Please also see the esorex documentation for naming of output products.	zpl_science_p1_minus_u_cam1.fits	-
zpl.science_p1.outfilename_q_cam2	string	The output filename for the final science product Q. Please also see the esorex documentation for naming of output products.	zpl_science_p1_q_cam2.fits	-
zpl.science_p1_plus_q_cam2.outfilename	string	The output filename for the science plus product +Q. Please also see the esorex documentation for naming of output products.	zpl_science_p1_plus_q_cam2.fits	-
zpl.science_p1_minus_q_cam2.outfilename	string	The output filename for the science minus product -Q. Please also see the esorex documentation for naming of output products.	zpl_science_p1_minus_q_cam2.fits	-
zpl.science_p1.outfilename_u_cam2	string	The output filename for the final science product U. Please also see the esorex documentation for naming of output products.	zpl_science_p1_u_cam2.fits	-
zpl.science_p1_plus_u_cam2.outfilename	string	The output filename for the science plus product +U. Please also see the esorex documentation for naming of output products.	zpl_science_p1_plus_u_cam2.fits	-

Name	Type	Description	Default	Allowed vals.
zpl.science_ p1_ minus_ u_ cam2.outfilename	string	The output filename for the science minus product -U. Please also see the esorex documentation for naming of output products.	zpl_ science_ p1_ minus_ u_ cam2.fits	-
zpl.science_ p1.subtract_ overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.science_ p1.keep_ intermediate	bool	Flag to set if intermediate date must be saved, namely pre-processed and overscan pre-processed subtracted data (FALSE)	0	-
zpl.science_ p1.save_ interprod	bool	Flag to set if the field center table must be saved as intermediate product (FALSE) Note that this parameter must be only applied for the offline pipeline	0	-
zpl.science_ p1.cut_ edge_ columns	bool	Flag to set if the first and last columns in the double images must be cut off in order to get square images. It happens in the case if double images are still not squared after the interpolation in the y-direction. If the flag is not set then the images will be squared by adding/copying top and bottom lines in the double images	0	-
zpl.science_ p1.coll_ alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median. Default is 0 = Mean.	0	0,1,2
zpl.science_ p1.filter_ radius	double	Filter radius for framecombination. A non zero value leads to suppression of high frequencies in the fourier domain before framecombination. The value expresses the minimum unsuppressed frequency as fraction of total frequency domain radius (a value of 1 would suppress essentially all frequencies).	0.0	0.0-1.0
zpl.preproc.outfilename_ cam1	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-1.	preproc_ cam1.fits	-
zpl.preproc.outfilename_ cam2	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-2.	preproc_ cam2.fits	-

Description:

The recipe produces combined science frame [and corresponing Mueller matrix elements (not implemented!)] of the Q and/or U measurements the in the polarization modes.The input frames might be either science polarimetric raw frames with the ZPL_SCIENCE_P1_RAW tag,or pre-processed science raw frames, which should carry the ZPL_SCIENCE_P1_PREPROC_CAM1 and/or ZPL_SCIENCE_P1_PREPROC_CAM2 tags, and calibration frames:- master bias calibration frames (if any) with the ZPL_MASTER_BIAS_CAM1 and/or ZPL_MASTER_BIAS_CAM2 tags, and - master dark calibration frames (if any) with the ZPL_MASTER_DARK_CAM1 and/or ZPL_MASTER_DARK_CAM2 tags, and- master intensity flat field calibration frames with the ZPL_INT_FLAT_FIELD_MASTER_CAM1 and/or ZPL_INT_FLAT_FIELD_MASTER_CAM2 tags, and- polariztion flat field calibration frames with the ZPL_POL_FLAT_PREPROC_CAM1 and/or ZPL_POL_FLAT_PREPROC_CAM2 tags, and- modem/de-modulation (modem) efficiency calibration frames with the ZPL_MODEM_EFF_CAM1 and/or ZPL_MODEM_EFF_CAM1 tags.The intensity flat calibration frames can be also used in the format of the QUAD IMAGE (see the description in sph_zpl_intensity_flat recipe) with the corresponding ZPL_INT_FLAT_FIELD_CAM1 and/or ZPL_INT_FLAT_FIELD_CAM2 tags. If both formats of the intensity flat field calibrations are presented in sof-file the MASTER format will be used. If input frames are raw frames then the polarization flat recipe first performs the pre-processing step for all input frames (raw cubes),creating corresponding pre-processed frames (cubes) for both ZIMPOL cameras (see also sph_zpl_preproc for the detailed description of the pre-processing step). Then, all pre-processed raw science frames are organized in the measurement groups with regards to the the stock parameters: Q [Qplus, Qminus] and/or or U [Uplus, Uminus]. These input frames frames should carry the SPH_ZPL_TAG_SCIENCE_P1_PREPROC_RAW tag.The pre-processed frames of each group for both cameras are then calibrated by subtacting a corresponding master bias frame and a master dark frame, and dividing the results by a corresponding intensity flat field frame. Then the stockes parameters are calulated for each group creating double image (I,P) frames.The polarization flat and modem efficiency calibrations are applied to the created double image frames of the stokes parameters. The calibrated frames of each group are then de-dithered, de-rotated and saved as intermediate products (note: if the zpl.science_p1.save_interprod is set to the 1, the recipe will also save the so called field center table which contains the the calculated center positions and parallactical angles for each plane of the pre-processed fits cube(s)). All de-dithered and de-rotated frames are avareged using collapse mean algorithm (for each group Qplus, Qminus, Uplus, Uminus).The combined frames of each groups of the DOUBLE IMAGE (8 extensions) format specified as follows:- combined intensity image (I), its badpixel-map, ncomb-map and rms-map.- combined polarimetric image (P), its badpixel-map, ncomb-map and rms-map.At the final step the double image frames (Qplus and Qminus) as well as (Uplus and Uminus) are combined polarimetrically (Q: I = [I(+Q) + I(-Q)]/2, P = [P(+Q) - P(-Q)]/2; U: I = [I(+U) + I(-U)]/2, P = [P(+U) - P(-U)]/2) The output Q and/or U double images for both cameras are reduced pipeline data products.

Products:

Name	Type	Description
ZPL_ SCIENCE_ P1_ REDUCED_ Q	FITS[Im(8)]	The final combined science frame [I_Q, P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the Q measurement I_Q corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the Q measurement P_Q , corresponding badpixel-map, ncomb-map and rms-map;

Name	Type	Description
ZPL_ SCIENCE_ P1_ REDUCED_ QPLUS	FITS[Im(8)]	The resulting combined science frame of [+I_Q, +P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science plus intensity image of the +Q measurement +I_Q, corresponding badpixel-map, ncomb-map and rms-map; - reduced science plus polarization image of the +Q measurement +P_Q, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P1_ REDUCED_ QMINUS	FITS[Im(8)]	The resulting combined science frame [-I_Q, -P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the -Q measurement -I_Q, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the -Q measurement -P_Q, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P1_ REDUCED_ U	FITS[Im(8)]	The final combined science frame [I_U, P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the U measurement I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the U measurement P_U, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P1_ REDUCED_ UPLUS	FITS[Im(8)]	The resulting combined science frame of [+I_U, +P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science plus intensity image of the +U measurement +I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science plus polarization image of the +U measurement +P_U, corresponding badpixel-map, ncomb-map and rms-map;

Name	Type	Description
ZPL_ SCIENCE_ P1_ REDUCED_ UMINUS	FITS[Im(8)]	The resulting combined science frame [-I_U, -P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the -U measurement -I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the -U measurement -P_U, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P1_ REDUCED_ Q_ CAM1	FITS[Im(8)]	The final combined science frame [I_Q, P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the Q measurement I_Q corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the Q measurement P_Q , corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P1_ REDUCED_ QPLUS_ CAM1	FITS[Im(8)]	The resulting combined science frame of [+I_Q, +P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science plus intensity image of the +Q measurement +I_Q, corresponding badpixel-map, ncomb-map and rms-map; - reduced science plus polarization image of the +Q measurement +P_Q, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P1_ REDUCED_ QMINUS_ CAM1	FITS[Im(8)]	The resulting combined science frame [-I_Q, -P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the -Q measurement -I_Q, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the -Q measurement -P_Q, corresponding badpixel-map, ncomb-map and rms-map;

Name	Type	Description
ZPL_ SCIENCE_ P1_ REDUCED_ U_ CAM1	FITS[Im(8)]	The final combined science frame [I_U, P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the U measurement I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the U measurement P_U, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P1_ REDUCED_ UPLUS_ CAM1	FITS[Im(8)]	The resulting combined science frame of [+I_U, +P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science plus intensity image of the +U measurement +I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science plus polarization image of the +U measurement +P_U, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P1_ REDUCED_ UMINUS_ CAM1	FITS[Im(8)]	The resulting combined science frame [-I_U, -P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the -U measurement -I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the -U measurement -P_U, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P1_ REDUCED_ Q_ CAM2	FITS[Im(8)]	The final combined science frame [I_Q, P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the Q measurement I_Q corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the Q measurement P_Q , corresponding badpixel-map, ncomb-map and rms-map;

Name	Type	Description
ZPL_ SCIENCE_ P1_ REDUCED_ QPLUS_ CAM2	FITS[Im(8)]	The resulting combined science frame of [+I_Q, +P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science plus intensity image of the +Q measurement +I_Q, corresponding badpixel-map, ncomb-map and rms-map; - reduced science plus polarization image of the +Q measurement +P_Q, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P1_ REDUCED_ QMINUS_ CAM2	FITS[Im(8)]	The resulting combined science frame [-I_Q, -P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the -Q measurement -I_Q, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the -Q measurement -P_Q, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P1_ REDUCED_ U_ CAM2	FITS[Im(8)]	The final combined science frame [I_U, P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the U measurement I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the U measurement P_U, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P1_ REDUCED_ UPLUS_ CAM2	FITS[Im(8)]	The resulting combined science frame of [+I_U, +P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science plus intensity image of the +U measurement +I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science plus polarization image of the +U measurement +P_U, corresponding badpixel-map, ncomb-map and rms-map;

Name	Type	Description
ZPL_ SCIENCE_ P1_ REDUCED_ UMINUS_ CAM2	FITS[Im(8)]	The resulting combined science frame [-I_U, -P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the -U measurement -I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the -U measurement -P_U, corresponding badpixel-map, ncomb-map and rms-map;

10.20 sph_zpl_science_p23

Purpose:

Reduce science frames of the Q and/or U observations for the polarization P2 and P3 modes.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ SCIENCE_ P23_ RAW	Raw data	Yes	0	Any
ZPL_ SCIENCE_ P23_ PREPROC_ CAM1	Calibration	Yes	0	Any
ZPL_ SCIENCE_ P23_ PREPROC_ CAM2	Calibration	Yes	0	Any
ZPL_ MASTER_ BIAS_ CAM1	Calibration	Yes	0	1
ZPL_ MASTER_ BIAS_ CAM2	Calibration	Yes	0	1
ZPL_ MASTER_ DARK_ CAM1	Calibration	Yes	0	1
ZPL_ MASTER_ DARK_ CAM2	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ CAM1	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ CAM2	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ MASTER_ CAM1	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ MASTER_ CAM2	Calibration	Yes	0	1
ZPL_ POL_ FLAT_ FIELD_ CAM1	Calibration	Yes	0	1
ZPL_ POL_ FLAT_ FIELD_ CAM2	Calibration	Yes	0	1
ZPL_ MODEM_ EFF_ CAM1	Calibration	Yes	0	1
ZPL_ MODEM_ EFF_ CAM2	Calibration	Yes	0	1
ZPL_ CENTER_ TABLE	Calibration	Yes	0	Any

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if the pre-processed data are used. As the format of the zimpol pre-processed data is complicated, this keyword was introduced in order to garantee that the pre-processed input frames are polarimetric pre-processed data, produced by the sph_zpl_preproc utility recipe. Note: if raw data are used (default), then all keywords needed for the pre-processing recipe (see sph_zpl_preproc) must be presented in the raw data.
ESO OCS3 ZIMPOL POL STOKES	string	No	Stokes parameters (Qplus, Qminus, Uplus, Uminus) SPH_COMMON_KEYWORD_CAM1_DITHERING_X double 0 0 100.0 X-position of the arm1(camera-1) [pix] SPH_COMMON_KEYWORD_CAM1_DITHERING_Y double 0 0 100.0 Y-position of the arm1(camera-1) [pix] SPH_COMMON_KEYWORD_CAM2_DITHERING_X double 0 0 100.0 X-position of the arm2(camera-2) [pix] SPH_COMMON_KEYWORD_CAM2_DITHERING_Y double 0 0 100.0 Y-position of the arm2(camera-2) [pix] SPH_COMMON_KEYWORD_DROT2_MODE string 0 0 0 De-rotator mode: ELEV(pupil stabilized), SKY(field stabilized)

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.science_ p23.outfilename_ q_ cam1	string	The output filename for the final science product Q. Please also see the esorex documentation for naming of output products.	zpl_ science_ p23_ q_ cam1.fits	-
zpl.science_ p23_ plus_ q_ cam1.outfilename	string	The output filename for the science plus product +Q. Please also see the esorex documentation for naming of output products.	zpl_ science_ p23_ plus_ q_ cam1.fits	-
zpl.science_ p23_ minus_ q_ cam1.outfilename	string	The output filename for the science minus product -Q. Please also see the esorex documentation for naming of output products.	zpl_ science_ p23_ minus_ q_ cam1.fits	-
zpl.science_ p23.outfilename_ u_ cam1	string	The output filename for the final science product U. Please also see the esorex documentation for naming of output products.	zpl_ science_ p23_ u_ cam1.fits	-
zpl.science_ p23_ plus_ u_ cam1.outfilename	string	The output filename for the science plus product +U. Please also see the esorex documentation for naming of output products.	zpl_ science_ p23_ plus_ u_ cam1.fits	-

Name	Type	Description	Default	Allowed vals.
zpl.science_ p23_ minus_ u_ cam1.outfilename	string	The output filename for the science minus product -U. Please also see the esorex documentation for naming of output products.	zpl_ science_ p23_ minus_ u_ cam1.fits	-
zpl.science_ p23.outfilename_ q_ cam2	string	The output filename for the final science product Q. Please also see the esorex documentation for naming of output products.	zpl_ science_ p23_ q_ cam2.fits	-
zpl.science_ p23_ plus_ q_ cam2.outfilename	string	The output filename for the science plus product +Q. Please also see the esorex documentation for naming of output products.	zpl_ science_ p23_ plus_ q_ cam2.fits	-
zpl.science_ p23_ minus_ q_ cam2.outfilename	string	The output filename for the science minus product -Q. Please also see the esorex documentation for naming of output products.	zpl_ science_ p23_ minus_ q_ cam2.fits	-
zpl.science_ p23.outfilename_ u_ cam2	string	The output filename for the final science product U. Please also see the esorex documentation for naming of output products.	zpl_ science_ p23_ u_ cam2.fits	-
zpl.science_ p23_ plus_ u_ cam2.outfilename	string	The output filename for the science plus product +U. Please also see the esorex documentation for naming of output products.	zpl_ science_ p23_ plus_ u_ cam2.fits	-
zpl.science_ p23_ minus_ u_ cam2.outfilename	string	The output filename for the science minus product -U. Please also see the esorex documentation for naming of output products.	zpl_ science_ p23_ minus_ u_ cam2.fits	-
zpl.science_ p23.subtract_ overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.science_ p23.save_ interprod	bool	Flag to set if the fiel center table must be saved as intermediate product (FALSE) Note that this parameter must be only applied for the offline pipeline	0	-
zpl.science_ p23.cut_ edge_ columns	bool	Flag to set if the first and last columns in the double images must be cut off in order to get square images. It happens in the case if double images are still not squared after the interpolation in the y-direction. If the flag is not set then the images will be squared by adding/copying top and bottom lines in the double images	0	-
zpl.science_ p23.coll_ alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median. Default is 0 = Mean.	0	0,1,2

Name	Type	Description	Default	Allowed vals.
zpl.science_ p23.filter_ radius	double	Filter radius for framecombination. A non zero value leads to suppression of high frequencies in the fourier domain before framecombination. The value expresses the minimum unsuppressed frequency as fraction of total frequency domain radius (a value of 1 would suppress essentially all frequencies).	0.0	0.0-1.0
zpl.preproc.outfilename_ cam1	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-1.	preproc_ cam1.fits	-
zpl.preproc.outfilename_ cam2	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-2.	preproc_ cam2.fits	-

Description:

The recipe produces combined science frame [and corresponing Mueller matrix elements (not implemented)] of the Q and/or U measurements the in the polarization modes.The input frames might be either science polarimetric raw frames with the ZPL_SCIENCE_P23_RAW tag,or pre-processed science raw frames, which should carry the ZPL_SCIENCE_P23_PREPROC_CAM1 and/or ZPL_SCIENCE_P23_PREPROC_CAM2 tags, and calibration frames:- master bias calibration frames (if any) with the ZPL_MASTER_BIAS_CAM1 and/or ZPL_MASTER_BIAS_CAM2 tags, and - master dark calibration frames (if any) with the ZPL_MASTER_DARK_CAM1 and/or ZPL_MASTER_DARK_CAM2 tags, and- master intensity flat field calibration frames with the ZPL_INT_FLAT_FIELD_MASTER_CAM1 and/or ZPL_INT_FLAT_FIELD_MASTER_CAM2 tags, and- polariztion flat field calibration frames with the ZPL_POL_FLAT_PREPROC_CAM1 and/or ZPL_POL_FLAT_PREPROC_CAM2 tags, and- modem/de-modulation (modem) efficiency calibration frames with the ZPL_MODEM_EFF_CAM1 and/or ZPL_MODEM_EFF_CAM1 tags.The intensity flat calibration frames can be also used in the format of the QUAD IMAGE (see the description in sph_zpl_intensity_flat recipe) with the corresponding ZPL_INT_FLAT_FIELD_CAM1 and/or ZPL_INT_FLAT_FIELD_CAM2 tags. If both formats of the intensity flat field calibrations are presented in sof-file the MASTER format will be used. If input frames are raw frames then the polarization flat recipe first performs the pre-processing step for all input frames (raw cubes),creating corresponding pre-processed frames (cubes) for both ZIMPOL cameras (see also sph_zpl_preproc for the detailed description of the pre-processing step). Then, all pre-processed raw science frames are organized in the measurement groups with regards to the the stock parameters: Q [Qplus, Qminus] and/or or U [Uplus, Uminus]. These input frames frames should carry the SPH_ZPL_TAG_SCIENCE_P23_PREPROC_RAW tag.The pre-processed frames of each group for both cameras are then calibrated by subtacting a corresponding master bias frame and a master dark frame, and dividing the results by a corresponding intensity flat field frame. Then the stockes parameters are calulated for each group creating double image (I,P) frames.The polarization flat and modem efficiency calibrations are applied to the created double image frames of the stokes parameters. The calibrated frames of each group are then de-dithered, de-rotated and saved as intermediate products (note: if the zpl.science_p23.save_interprod is set to the 1, the recipe will also save the so called field center table which contains the the calculated center positions for each plane of the pre-processed fits cube(s)). All de-dithered frames are avareged using collapse mean algorithm (for each group Qplus, Qminus, Uplus, Uminus).The combined frames of each groups of the DOUBLE IMAGE (8 extensions) format specified as follows:- combined intensity image (I), its badpixel-map, ncomb-map and rms-map.- combined polarimetric image (P), its badpixel-map, ncomb-map and rms-map.At the final step the double image frames (Qplus and Qminus) as well as (Uplus and Uminus) are combined polarimetrically (Q: I = [I(+Q) + I(-Q)]/2, P = [P(+Q) - P(-Q)]/2; U: I = [I(+U) + I(-U)]/2, P = [P(+U) - P(-U)]/2) The output Q and/or U double images for both cameras are reduced pipeline data products.

Products:

Name	Type	Description
ZPL_ SCIENCE_ P23_ REDUCED_ Q_ CAM1	FITS[Im(8)]	The final combined science frame [I_Q, P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the Q measurement I_Q corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the Q measurement P_Q , corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P23_ REDUCED_ QPLUS_ CAM1	FITS[Im(8)]	The resulting combined science frame of [+I_Q, +P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science plus intensity image of the +Q measurement +I_Q, corresponding badpixel-map, ncomb-map and rms-map; - reduced science plus polarization image of the +Q measurement +P_Q, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P23_ REDUCED_ QMINUS_ CAM1	FITS[Im(8)]	The resulting combined science frame [-I_Q, -P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the -Q measurement -I_Q, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the -Q measurement -P_Q, corresponding badpixel-map, ncomb-map and rms-map;

Name	Type	Description
ZPL_ SCIENCE_ P23_ REDUCED_ U_ CAM1	FITS[Im(8)]	The resulting combined science frame [I_U, P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the U measurement I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the U measurement P_U, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P23_ REDUCED_ UPLUS_ CAM1	FITS[Im(8)]	The resulting combined science frame of [+I_U, +P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science plus intensity image of the +U measurement +I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science plus polarization image of the +U measurement +P_U, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P23_ REDUCED_ UMINUS_ CAM1	FITS[Im(8)]	The resulting combined science frame [-I_U, -P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the -U measurement -I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the -U measurement -P_U, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P23_ REDUCED_ Q_ CAM2	FITS[Im(8)]	The resulting combined science frame [I_Q, P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the Q measurement I_Q corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the Q measurement P_Q , corresponding badpixel-map, ncomb-map and rms-map;

Name	Type	Description
ZPL_ SCIENCE_ P23_ REDUCED_ QPLUS_ CAM2	FITS[Im(8)]	The resulting combined science frame of [+I_Q, +P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science plus intensity image of the +Q measurement +I_Q, corresponding badpixel-map, ncomb-map and rms-map; - reduced science plus polarization image of the +Q measurement +P_Q, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P23_ REDUCED_ QMINUS_ CAM2	FITS[Im(8)]	The resulting combined science frame [-I_Q, -P_Q] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the -Q measurement -I_Q, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the -Q measurement -P_Q, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P23_ REDUCED_ U_ CAM2	FITS[Im(8)]	The resulting combined science frame [I_U, P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the U measurement I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the U measurement P_U, corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ SCIENCE_ P23_ REDUCED_ UPLUS_ CAM2	FITS[Im(8)]	The resulting combined science frame of [+I_U, +P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science plus intensity image of the +U measurement +I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science plus polarization image of the +U measurement +P_U, corresponding badpixel-map, ncomb-map and rms-map;

Name	Type	Description
ZPL_ SCIENCE_ P23_ REDUCED_ UMINUS_ CAM2	FITS[Im(8)]	The resulting combined science frame [-I_U, -P_U] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - reduced science intensity image of the -U measurement -I_U, corresponding badpixel-map, ncomb-map and rms-map; - reduced science polarization image of the -U measurement -P_U, corresponding badpixel-map, ncomb-map and rms-map;

10.21 sph_zpl_science_imaging

Purpose:

Reduce science frames in the imaging modes.

Type:

Technical calibration

Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ SCIENCE_ IMAGING_ RAW	Raw data	Yes	0	Any
ZPL_ SCIENCE_ IMAGING_ PREPROC_ RAW	Raw data	Yes	0	Any
ZPL_ SCIENCE_ IMAGING_ PREPROC_ CAM1	Calibration	Yes	0	Any
ZPL_ SCIENCE_ IMAGING_ PREPROC_ CAM2	Calibration	Yes	0	Any
ZPL_ MASTER_ BIAS_ IMAGING	Calibration	Yes	0	1
ZPL_ MASTER_ BIAS_ IMAGING_ CAM1	Calibration	Yes	0	1
ZPL_ MASTER_ BIAS_ IMAGING_ CAM2	Calibration	Yes	0	1
ZPL_ MASTER_ DARK_ IMAGING	Calibration	Yes	0	1
ZPL_ MASTER_ DARK_ IMAGING_ CAM1	Calibration	Yes	0	1
ZPL_ MASTER_ DARK_ IMAGING_ CAM2	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ IMAGING	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ IMAGING_ CAM1	Calibration	Yes	0	1
ZPL_ INT_ FLAT_ FIELD_ IMAGING_ CAM2	Calibration	Yes	0	1
ZPL_ CENTER_ TABLE	Calibration	Yes	0	Any

Raw frame keywords used:

Keyword	Type	Optional	Description
ESO DRS PC PROD TYPE	string	No	This keyword is mandatory if the pre-processed data are used. As the format of the zimpol data is complicated, this keyword is introduced in order to garantee that the input frames are imaging pre-processed data, produced by the sph_zpl_preproc_imaging recipe which added this keyword automatically. The value of this keyword is set up to >>SPH PC PREPROC ZPL EXP IMAGING<<. Note: if raw data are used (default), then all keywords needed for the pre-processing recipe (see sph_zpl_preproc_imaging) must be presented in the raw data. SPH_COMMON_KEYWORD_CAM1_DITHERING_X double 0 0 100.0 X-position of the arm1(camera-1) [pix] SPH_COMMON_KEYWORD_CAM1_DITHERING_Y double 0 0 100.0 Y-position of the arm1(camera-1) [pix] SPH_COMMON_KEYWORD_CAM2_DITHERING_X double 0 0 100.0 X-position of the arm2(camera-2) [pix] SPH_COMMON_KEYWORD_CAM2_DITHERING_Y double 0 0 100.0 Y-position of the arm2(camera-2) [pix] SPH_COMMON_KEYWORD_DROT2_MODE string 0 0 0 De-rotator mode: ELEV(pupil stabilized), SKY(field stabilized)

Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.science_imaging.outfilename	string	The output filename for the product, camera-1/2. Please also see the esorex documentation for naming of output products.	zpl_science_imaging.fits	-
zpl.science_imaging.outfilename_cam1	string	The output filename for the product, camera-1. Please also see the esorex documentation for naming of output products.	zpl_science_imaging_cam1.fits	-
zpl.science_imaging.outfilename_cam2	string	The output filename for the product, camera2. Please also see the esorex documentation for naming of output products.	zpl_science_imaging_cam2.fits	-
zpl.science_p1.subtract_overscan	bool	Flag to set if the overscan mean values must be subtracted from pre-processed data (TRUE) Note that this parameter is applied if pre-processed data containt overscan table	1	-
zpl.science_imaging.coll_alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median. Default is 0 = Mean.	0	0,1

Name	Type	Description	Default	Allowed vals.
zpl.science_ imaging.filter_ radius	double	Filter radius for the framecombination. A non zero value leads to suppression of high frequencies in the fourier domain before framecombination. The value expresses the minimum unsuppressed frequency as fraction of total frequency domain radius (a value of 1 would suppress essentially all frequencies).	0.0	0.0-1.0
zpl.science_ imaging.keep_ intermediate	bool	Flag to set if intermediate date must be saved, namely pre-processed and overscan pre-processed subtracted data, linbadpix map and non-normalized products (FALSE)	0	-
zpl.science_ imaging.save_ interprod	bool	Flag to set if the fiel center table must be saved as intermediate product (FALSE) Note that this parameter must be only applied for the offline pipeline	0	-
zpl.science_ imaging.cut_ edge_ columns	bool	Flag to set if the first and last columns in the double images must be cut off in order to get square images. It happens in the case if double images are still not squared after the interpolation in the y-direction. If the flag is not set then the images will be squared by adding/copying top and bottom lines in the double images	0	-
zpl.preproc.outfilename_ cam1	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-1.	preproc_ cam1.fits	-
zpl.preproc.outfilename_ cam2	string	The postfix- of the intermediate filename of the pre-processed raw data for the CAMERA-2.	preproc_ cam2.fits	-

Description:

The recipe reduces combined science frames for the imaging modes.The input frames might be either science imaging raw frames with the ZPL_SCIENCE_IMAGING_RAW tag or pre-processed science imaging frames, which should carry the ZPL_SCIENCE_IMAGING_PREPROC_CAM1 and/orZPL_SCIENCE_IMAGING_PREPROC_CAM2 tags, and master bias frames (if any) with the ZPL_MASTER_BIAS_IMAGING_CAM1 and/or ZPL_MASTER_BIAS_IMAGING_CAM2 tags, and and master dark frames (if any) with the ZPL_MASTER_DARK_IMAGING_CAM1 and/or ZPL_MASTER_DARK_IMAGING_CAM2 tags, and intensity flat field frames (if any) with the ZPL_INT_FLAT_IMAGING_CAM1 and/or ZPL_INT_FLAT_IMAGING_CAM2 tags.If input frames are raw frames, the science imaging recipe first performs the pre-processing step for all input frames (raw cubes),creating corresponding pre-processed frames (cubes) for both ZIMPOL cameras (see also sph_zpl_preproc_imaging for the detailed description of the pre-processing step). The pre-processed imaging science frames are then calibrated (for each camera) by subtracting the master imaging bias and the master imaging dark, and divided by the corresponding intensity imaging flat field. The calibrated frames are de-dithered and de-rotated and then saved as intermediate products. The final step is to combine all these calibrated, de-dithered and de-rotated frames, using a standard mean algorithm.The combined frames for both cameras are of the DOUBLE IMAGE (8 extensions) format specified as follows:- combined intensity science image, badpixel-map, ncomb-map and rms-map.- combined science dark image (dark current), badpixel-map, ncomb-map and rms-map.The output double image frames are reduced pipeline data products for both cameras.

Products:

Name	Type	Description
ZPL_ SCIENCE_ IMAGING_ REDUCED	FITS[Im(8)]	The resulting reduced science imaging frame of the DOUBLE IMAGING format for the camera-1. This frame contains 8 image extensions: - science intensity image, badpixel-map, ncomb-map and rms-map; - science dark image (dark current), badpixel-map, ncomb-map and rms-map.
ZPL_ SCIENCE_ IMAGING_ REDUCED_ CAM1	FITS[Im(8)]	The resulting reduced science imaging frame of the DOUBLE IMAGING format for the camera-1. This frame contains 8 image extensions: - science intensity image, badpixel-map, ncomb-map and rms-map; - science dark image (dark current), badpixel-map, ncomb-map and rms-map.
ZPL_ SCIENCE_ IMAGING_ REDUCED_ CAM2	FITS[Im(8)]	The resulting reduced science imaging frame of the DOUBLE IMAGING format for the camera-2. This frame contains 8 image extensions: - science intensity image, badpixel-map, ncomb-map and rms-map; - science dark image (dark current), badpixel-map, ncomb-map and rms-map.

Chapter 11

Installation Procedure and Troubleshooting

11.1 Installing the Pipeline

To install the pipeline, please follow the following instructions:

Once you have downloaded (from http://www.mpia-hd.mpg.de/SPHERE/Releases/sphere-kit-0.13.0.tar.gz) or gotten otherwise a tarball unpack it and cd to the main directory

```
tar -xvzf sphere-kit-0.13.0.tar.gz
cd sphere-kit-0.13.0
```

In the directory you will find a script called install_pipeline. Run it with

```
./install_pipeline
```

It will prompt you for a directory to install the pipeline in. We recommend choosing as name /home/myself/sphereP where home/myself is your home directory. All software should now be installed automatically. This will take a while.

A special note for mac users: if you have a mac, there is currently no support for the SPHERE pipeline. However, with some small changes it is possible to install the pipeline on a mac. Please see http://www.mpia.de/SPHERE/WIKI/pmwiki.php?n=SPHEREInstallation.Installation.

When everything is done, and you are alright with waiting for a while, you may do the following to make sure it all worked: in the sphere-kit directory (from the unpacked tarball) type

```
cd sphere-0.13.0
make check
```

and hopefully it should end with a statement "All <a number> tests passed" or so. But be warned that this may take quite a long time ! If it fails, please do contact us (moeller@mpia-hd.mpg.de or pavlov@mpia-hd.mpg.de).

Setting everything up to run recipes with esorex

To run recipes with esorex you first need to set some environment variables. How you do this in detail depends on your shell. In the description here we assume you have bash. To set the environment variables, add this to your .bashrc file:

```
export PATH=/home/me/sphereP/bin:$PATH
export ESOREX_PLUGIN_DIR=/home/me/sphereP/lib/esopipes-plugins/sphere-0.13.0:
                          $ESOREX_PLUGIN_DIR
```

Here you should substitute /home/me/sphereP with the directory where you installed everything (the name you gave when you ran the install_pipeline script). If you have a apple Mac, change LD_LIBRAY_PATH to DYLD_LIBRARY_PATH. To test it, open a new xterm and type

```
esorex --recipes
```

and it should give

```
***** ESO Recipe Execution Tool, version 3.9.0 *****
List of Available Recipes :
sph_ifs_master_detector_flat : Creates the master detector flat
                               from a list of input frames
sph_ifs_master_dark : Creates the master dark from a list of
                      input dark frames
```

For more info on ESOREX, please see ESOrex's homepage (http://www.eso.org/sci/data-processing/software/cpl/esorex.html) or just type

```
esorex --help
```

11.2 Tips, tricks, and troubleshooting

11.2.1 My recipe run terminats with thousands of error messages...

Don't panic! Try to go through the log file produced by esorex and find the first ERROR signal. In many cases, this gives a hint what went wrong and what can be tried to correct the behaviour...

11.2.2 I still don't understand the error / it says that the 'actual error was lost'

Now panic! Better still, note that many parts of the pipeline are still untested and it's not unlikely you stumbled onto something that has never been tried before. Try to make a small package of data and a commandline that can reproduce the eroror and contact us!

11.2.3 My distortion map or other peak-finding involving recipe doesn't produce an output and gives errors.

Recipes involving peak finding usually offer a `threshold` or `sigma` parameter in the command line. Some also offer `nsources` to pre-determine the number of sources to be found. Try playing with these in conjunction to looking at the images, in particular the peak intensity in relation to the background level, which should be `threshold` or `sigma` times apart. Note than in combination with a supplied as e.g. in sph_ird_star_center, the parameter value needed can be rather particular... so try also very large numbers here!

Appendix A

Quality Control Keywords

A.1 Common

```
#
# Code define: SPH_COMMON_KEYWORD_QC_MEANMASTERFRAME
#
#    Code references:
#     - sph_master_frame.c
#
Parameter Name:    ESO QC MEANMASTER
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Mean value of a master frame
Description:       Mean value of a master frame

#
# Code define: SPH_COMMON_KEYWORD_QC_MEDIANMASTERFRAME
#
#    Code references:
#     - sph_master_frame.c
#
Parameter Name:    ESO QC MEDIANMASTER
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Median value of a master frame
Description:       Median value of a master frame

#
# Code define: SPH_COMMON_KEYWORD_QC_RMSMASTERFRAME
#
#    Code references:
#     - sph_master_frame.c
#
Parameter Name:    ESO QC RMSMASTER
```

```
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      RMS of a master frame
Description:        RMS of a master frame

#
# Code define: SPH_COMMON_KEYWORD_QC_RON
#
#    Code references:
#      - sph_gain_and_ron.c
#      - sph_ifs_gain_run.c
#      - sph_ird_gain_run.c
#
Parameter Name:     ESO QC RON
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Determined read-out noise (ADU)
Description:        Determined read-out noise (ADU)

#
# Code define: SPH_COMMON_KEYWORD_QC_RON_RMS
#
#    Code references:
#      - sph_gain_and_ron.c
#      - sph_ifs_gain_run.c
#      - sph_ird_gain_run.c
#
Parameter Name:     ESO QC RON RMS
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      RMS of determined read-outnoise (ADU)
Description:        RMS of determined read-out noise (ADU)

#
# Code define: SPH_COMMON_KEYWORD_QC_GAIN
#
#    Code references:
#      - sph_gain_and_ron.c
#      - sph_ifs_gain_run.c
#      - sph_ird_gain_run.c
#
Parameter Name:     ESO QC GAIN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Determined gain (e-/ADU)
Description:        Determined gain (e-/ADU)
```

```
#
# Code define: SPH_COMMON_KEYWORD_QC_GAIN_RMS
#
#    Code references:
#     - sph_gain_and_ron.c
#     - sph_ifs_gain_run.c
#     - sph_ird_gain_run.c
#
Parameter Name:    ESO QC GAIN RMS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Determined RMS of gain (e-/ADU)
Description:       Determined RMS of gain (e-/ADU)

#
# Code define: SPH_COMMON_KEYWORD_QC_GAIN_SIGNAL
#
#
Parameter Name:    ESO QC MON SIGNAL
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Comment Field:     Determined RMS of gain (e-/ADU)
Description:
#
# Code define: SPH_COMMON_KEYWORD_QC_GAIN_SQ_NOISE
#
#
Parameter Name:    ESO QC MON NOISE
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Comment Field:     Determined RMS of gain (e-/ADU)
Description:
#
# Code define: SPH_COMMON_KEYWORD_QC_PERSISTENCE_ONES
#
#    Code references:
#     - sph_irdifs_persistence.c
#
Parameter Name:    ESO QC PERSISTENCE 1S
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     saturation signal remainder after 1s
Description:       saturation signal remainder after 1s

#
```

```
# Code define: SPH_COMMON_KEYWORD_QC_PERSISTENCE_FIVES
#
#    Code references:
#     - sph_irdifs_persistence.c
#
Parameter Name:    ESO QC PERSISTENCE 5S
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     saturation signal remainder after 5s
Description:       saturation signal remainder after 5s

#
# Code define: SPH_COMMON_KEYWORD_QC_PERSISTENCE_TENS
#
#    Code references:
#     - sph_irdifs_persistence.c
#
Parameter Name:    ESO QC PERSISTENCE 10S
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     saturation signal remainder after 10s
Description:       saturation signal remainder after 10s

#
# Code define: SPH_COMMON_KEYWORD_QC_PERSISTENCE_RMS
#
#
Parameter Name:    ESO QC PERSISTENCE RMS
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Comment Field:     RMS of saturation signal
Description:       RMS of saturation signal

#
# Code define: SPH_COMMON_KEYWORD_QC_PERSISTENCE_COEFF
#
#    Code references:
#     - sph_irdifs_persistence.c
#
Parameter Name:    ESO QC PERSISTENCE COEFF
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     persistence coefficient
Description:       persistence coefficient

#
```

```
# Code define: SPH_COMMON_KEYWORD_QC_PERSISTENCE_COEFF_RMS
#
#    Code references:
#     - sph_irdifs_persistence.c
#
Parameter Name:    ESO QC PERSISTENCE COEFF RMS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     RMS of persistence coefficient
Description:       RMS of persistence coefficient

#
# Code define: SPH_COMMON_KEYWORD_QC_MEAN_DOUBLEIMAGE_IFRAME
#
#    Code references:
#     - sph_double_image.c
#
Parameter Name:    ESO QC DOUBLE IMAGE IFRAME MEAN
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Mean value of an intensity double image (ZIMPOL)
Description:       Mean value of an intensity double image (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEDIAN_DOUBLEIMAGE_IFRAME
#
#    Code references:
#     - sph_double_image.c
#
Parameter Name:    ESO QC DOUBLE IMAGE IFRAME MEDIAN
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Median value of an intensity double image (ZIMPOL)
Description:       Median value of an intensity double image (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_RMS_DOUBLEIMAGE_IFRAME
#
#    Code references:
#     - sph_double_image.c
#
Parameter Name:    ESO QC DOUBLE IMAGE IFRAME RMS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     RMS of an intensity double image (ZIMPOL)
Description:       RMS of an intensity double image (ZIMPOL)
```

```
#
# Code define: SPH_COMMON_KEYWORD_QC_MEAN_DOUBLEIMAGE_PFRAME
#
#    Code references:
#     - sph_double_image.c
#
Parameter Name:     ESO QC DOUBLE IMAGE PFRAME MEAN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Mean value of a polarization degree double image (ZIMPOL)
Description:        Mean value of a polarization degree double image (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEDIAN_DOUBLEIMAGE_PFRAME
#
#    Code references:
#     - sph_double_image.c
#
Parameter Name:     ESO QC DOUBLE IMAGE PFRAME MEDIAN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Median value of a polarization degree double image (ZIMPOL)
Description:        Medan value of a polarization degree double image (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_RMS_DOUBLEIMAGE_PFRAME
#
#    Code references:
#     - sph_double_image.c
#
Parameter Name:     ESO QC DOUBLE IMAGE PFRAME RMS
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      RMS of a polarization degree double image (ZIMPOL)
Description:        RMS of a polarization degree double image (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEAN_TRIPLEIMAGE_IFRAME
#
#    Code references:
#     - sph_triple_image.c
#
Parameter Name:     ESO QC TRIPLE IMAGE IFRAME MEAN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
```

```
Comment Field:     Mean value of an intensity triple image (ZIMPOL)
Description:       Mean value of an intensity triple image (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEDIAN_TRIPLEIMAGE_IFRAME
#
#    Code references:
#     - sph_triple_image.c
#
Parameter Name:    ESO QC TRIPLE IMAGE IFRAME MEDIAN
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Median value of an intensity triple image (ZIMPOL)
Description:       Median value of an intensity triple image (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_RMS_TRIPLEIMAGE_IFRAME
#
#    Code references:
#     - sph_triple_image.c
#
Parameter Name:    ESO QC TRIPLE IMAGE IFRAME RMS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     RMS of an intensity triple image (ZIMPOL)
Description:       RMS of an intensity triple image (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEAN_TRIPLEIMAGE_QFRAME
#
#    Code references:
#     - sph_triple_image.c
#
Parameter Name:    ESO QC TRIPLE IMAGE QFRAME MEAN
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Mean value of a Q triple image (ZIMPOL)
Description:       Mean value of a Q triple image (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEDIAN_TRIPLEIMAGE_QFRAME
#
#    Code references:
#     - sph_triple_image.c
#
Parameter Name:    ESO QC TRIPLE IMAGE QFRAME MEDIAN
Class:             header|qc-log
Context:           process
```

```
Type:               double
Value Format:       %.2f
Comment Field:      Median value of a Q triple image (ZIMPOL)
Description:        Median value of a Q triple image (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_RMS_TRIPLEIMAGE_QFRAME
#
#    Code references:
#     - sph_triple_image.c
#
Parameter Name:     ESO QC TRIPLE IMAGE QFRAME RMS
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      RMS of a Q triple image (ZIMPOL)
Description:        RMS of a Q triple image (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEAN_TRIPLEIMAGE_UFRAME
#
#    Code references:
#     - sph_triple_image.c
#
Parameter Name:     ESO QC TRIPLE IMAGE UFRAME MEAN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Mean value of a U triple image (ZIMPOL)
Description:        Mean value of a U triple image (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEDIAN_TRIPLEIMAGE_UFRAME
#
#    Code references:
#     - sph_triple_image.c
#
Parameter Name:     ESO QC TRIPLE IMAGE UFRAME MEDIAN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Median value of a U triple image (ZIMPOL)
Description:        Median value of a U triple image (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_RMS_TRIPLEIMAGE_UFRAME
#
#    Code references:
#     - sph_triple_image.c
#
Parameter Name:     ESO QC TRIPLE IMAGE UFRAME RMS
```

```
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      RMS a U triple image (ZIMPOL)
Description:        RMS of a U triple image (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEAN_QUADIMAGE_ZERO_ODD
#
#    Code references:
#     - sph_quad_image.c
#
Parameter Name:     ESO QC QUAD IMAGE ZERO ODD MEAN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Mean value of a zero-odd quadimage
Description:        Mean value of a zero-odd quadimage (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEDIAN_QUADIMAGE_ZERO_ODD
#
#    Code references:
#     - sph_quad_image.c
#
Parameter Name:     ESO QC QUAD IMAGE ZERO ODD MEDIAN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Median value of a zero-odd quadimage
Description:        Median value of a zero-odd quadimage (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_RMS_QUADIMAGE_ZERO_ODD
#
#    Code references:
#     - sph_quad_image.c
#
Parameter Name:     ESO QC QUAD IMAGE ZERO ODD RMS
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      RMS of a zero-odd quadimage
Description:        RMS of a zero-odd quadimage (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEAN_QUADIMAGE_ZERO_EVEN
#
#    Code references:
#     - sph_quad_image.c
```

```
#
Parameter Name:    ESO QC QUAD IMAGE ZERO EVEN MEAN
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Mean value of a zero-even quadimage
Description:       Mean value of a zero-even quadimage (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEDIAN_QUADIMAGE_ZERO_EVEN
#
#    Code references:
#     - sph_quad_image.c
#
Parameter Name:    ESO QC QUAD IMAGE ZERO EVEN MEDIAN
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Median value of a zero-even quadimage
Description:       Median value of a zero-even quadimage (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_RMS_QUADIMAGE_ZERO_EVEN
#
#    Code references:
#     - sph_quad_image.c
#
Parameter Name:    ESO QC QUAD IMAGE ZERO EVEN RMS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     RMS of a zero-even quadimage
Description:       RMS of a zero-even quadimage (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEAN_QUADIMAGE_PI_ODD
#
#    Code references:
#     - sph_quad_image.c
#
Parameter Name:    ESO QC QUAD IMAGE PI ODD MEAN
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Mean value of a pi-odd quadimage
Description:       Mean value of a pi-odd quadimage (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEDIAN_QUADIMAGE_PI_ODD
#
```

```
#    Code references:
#     - sph_quad_image.c
#
Parameter Name:     ESO QC QUAD IMAGE PI ODD MEDIAN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Median value of a pi-odd quadimage
Description:        Median value of a pi-odd quadimage (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_RMS_QUADIMAGE_PI_ODD
#
#    Code references:
#     - sph_quad_image.c
#
Parameter Name:     ESO QC QUAD IMAGE PI ODD RMS
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      RMS of a pi-odd quadimage
Description:        RMS of a pi-odd quadimage (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEAN_QUADIMAGE_PI_EVEN
#
#    Code references:
#     - sph_quad_image.c
#
Parameter Name:     ESO QC QUAD IMAGE PI EVEN MEAN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Mean value of a pi-evn quadimage
Description:        Mean value of a pi-even quadimage (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_QC_MEDIAN_QUADIMAGE_PI_EVEN
#
#    Code references:
#     - sph_quad_image.c
#
Parameter Name:     ESO QC QUAD IMAGE PI EVEN MEDIAN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Median value of a pi-evn quadimage
Description:        Median value of a pi-even quadimage (ZIMPOL)

#
```

```
# Code define: SPH_COMMON_KEYWORD_QC_RMS_QUADIMAGE_PI_EVEN
#
#    Code references:
#     - sph_quad_image.c
#
Parameter Name:    ESO QC QUAD IMAGE PI EVEN RMS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     RMS of a pi-evn quadimage
Description:       RMS of a pi-even quadimage (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_DARK_MEAN_RON
#
#    Code references:
#     - sph_ifs_master_dark_run.c
#
Parameter Name:    ESO QC MEAN RON
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     mean dark RON estimate (ADU)
Description:       Mean dark RON estimate (ADU)

#
# Code define: SPH_COMMON_KEYWORD_NUMBER_HOTPIXELS
#
#    Code references:
#     - sph_ifs_master_dark_run.c
#     - sph_ird_master_dark_run.c
#
Parameter Name:    ESO QC NUM HOTPIXELS
Class:             header|qc-log
Context:           process
Type:              int
Value Format:      %d
Comment Field:     nunmber of identified hot pixels
Description:       Number of identified hot pixels

#
# Code define: SPH_COMMON_KEYWORD_FLAT_FPN
#
#    Code references:
#     - sph_ifs_master_detector_flat_run.c
#     - sph_ifs_instrument_flat_run.c
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:    ESO QC FPN
Class:             header|qc-log
Context:           process
Type:              double
```

```
Value Format:       %.2f
Comment Field:      Fixed pattern noise
Description:        Fixed pattern noise

#
# Code define: SPH_COMMON_KEYWORD_FLAT_RMS
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC RMS
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Root mean squared
Description:        Root mean squared

#
# Code define: SPH_COMMON_KEYWORD_FLAT_NONLIN_FACTOR
#
#    Code references:
#     - sph_ifs_master_detector_flat_run.c
#     - sph_ifs_instrument_flat_run.c
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC FLAT NONLINEARITY
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      nonlinearity coefficient
Description:        nonlinearity coefficient

#
# Code define: SPH_COMMON_KEYWORD_FLAT_MEAN_COUNT
#
#    Code references:
#     - sph_ifs_master_detector_flat_run.c
#     - sph_ifs_instrument_flat_run.c
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC FLAT MEAN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Mean counts in flat field
Description:        Mean counts in flat field

#
# Code define: SPH_COMMON_KEYWORD_NUMBER_BADPIXELS
#
#    Code references:
```

```
#     - sph_gain_and_ron.c
#     - sph_ifs_master_detector_flat_run.c
#     - sph_ifs_instrument_flat_run.c
#     - sph_zpl_intensity_flat_run.c
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:    ESO QC NUM BADPIXELS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     number of identified bad pixels
Description:       number of identified bad pixels

#
# Code define: SPH_COMMON_KEYWORD_FLAT_LAMP_FLUX
#
#    Code references:
#     - sph_ifs_master_detector_flat_run.c
#     - sph_ifs_instrument_flat_run.c
#     - sph_ifs_wave_calib_run.c
#
Parameter Name:    ESO QC LAMP FLUX AVG
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Lamp flux (counts/s in single frame)
Description:       Lamp flux (counts/s in single frame)

#
# Code define: SPH_COMMON_KEYWORD_FLAT_LAMP_FLUX_STDEV
#
#    Code references:
#     - sph_ifs_master_detector_flat_run.c
#     - sph_ifs_instrument_flat_run.c
#     - sph_ifs_wave_calib_run.c
#
Parameter Name:    ESO QC LAMP FLUX VARIANCE
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Lamp flux stdev (counts/s in single frame)
Description:       Lamp flux stdev (counts/s in single frame)

#
# Code define: SPH_COMMON_KEYWORD_MEDIAN_RESOLVING_POWER
#
#    Code references:
#     - sph_ifs_wave_calib_run.c
#
Parameter Name:    ESO QC MEDIAN RESOLVING POWER
Class:             header|qc-log
```

```
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      median resolving power
Description:        median resolving power

#
# Code define: SPH_COMMON_KEYWORD_MEDIAN_DISPERSION
#
#    Code references:
#     - sph_ifs_wave_calib_run.c
#
Parameter Name:     ESO QC MEDIAN DISPERSION
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      median value of longest wavelength
Description:        median value of longest wavelength

#
# Code define: SPH_COMMON_KEYWORD_MEDIAN_MAXWAVEL
#
#    Code references:
#     - sph_ifs_wave_calib_run.c
#
Parameter Name:     ESO QC MEDIAN MAX WAVEL
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      median value of longest wavelength
Description:        median value of longest wavelength

#
# Code define: SPH_COMMON_KEYWORD_QC_WAVE_NDISP
#
#    Code references:
#     - sph_ifs_wave_calib_run.c
#
Parameter Name:     ESO QC NUM OUT OF DISP
Class:              header|qc-log
Context:            process
Type:               int
Value Format:       %d
Comment Field:      no. of wavelengths out of 1 dispersion
Description:        no. of wavelengths out of 1 dispersion

#
# Code define: SPH_COMMON_KEYWORD_QC_WAVE_BADSPEC
#
#    Code references:
#     - sph_ifs_wave_calib_run.c
#
```

```
Parameter Name:    ESO QC BAD SPECTRA
Class:             header|qc-log
Context:           process
Type:              int
Value Format:      %d
Comment Field:     no. of bad spectra
Description:       no. of bad spectra

#
# Code define: SPH_COMMON_KEYWORD_DISTMAP_NREMOVED
#
#    Code references:
#     - sph_polynomial_distortion_model.c
#
Parameter Name:    ESO QC DISTMAP NREMOVED
Class:             header|qc-log
Context:           process
Type:              int
Value Format:      %d
Comment Field:     no. of points removed from distortion map
Description:       no. of points removed from distortion map

#
# Code define: SPH_COMMON_KEYWORD_DISTMAP_PIXSCALE
#
#    Code references:
#     - sph_polynomial_distortion_model.c
#
Parameter Name:    ESO QC DISTMAP PIXSCALE
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     distortion map pixel scale
Description:       distortion map pixel scale

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_ZPLEXP_ZERO_ODD
#
#    Code references:
#     - sph_zpl_intensity_flat_run.c
#
Parameter Name:    ESO QC ZPL EXP ZERO ODD NUMBER BADPIXELS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     number of bad pixels
Description:       number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_ZPLEXP_ZERO_EVEN
#
#    Code references:
```

```
#      - sph_zpl_intensity_flat_run.c
#
Parameter Name:     ESO QC ZPL EXP ZERO EVEN NUMBER BADPIXELS
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      number of bad pixels
Description:        number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_ZPLEXP_PI_ODD
#
#     Code references:
#      - sph_zpl_intensity_flat_run.c
#
Parameter Name:     ESO QC ZPL EXP PI ODD NUMBER BADPIXELS
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      number of bad pixels
Description:        number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_ZPLEXP_PI_EVEN
#
#     Code references:
#      - sph_zpl_intensity_flat_run.c
#
Parameter Name:     ESO QC ZPL EXP PI EVEN NUMBER BADPIXELS
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      number of bad pixels
Description:        number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_QUAD_IMAGE_ZERO_ODD
#
#     Code references:
#      - sph_zpl_master_bias_run.c
#      - sph_zpl_master_dark_run.c
#      - sph_zpl_intensity_flat_run.c
#      - sph_quad_image.c
#
Parameter Name:     ESO QC QUAD IMAGE ZERO ODD NUMBER BADPIXELS
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      number of bad pixels
Description:        number of bad pixels
```

```
#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_QUAD_IMAGE_ZERO_EVEN
#
#    Code references:
#     - sph_zpl_master_bias_run.c
#     - sph_zpl_master_dark_run.c
#     - sph_zpl_intensity_flat_run.c
#     - sph_quad_image.c
#
Parameter Name:    ESO QC QUAD IMAGE ZERO EVEN NUMBER BADPIXELS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     number of bad pixels
Description:       number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_QUAD_IMAGE_PI_ODD
#
#    Code references:
#     - sph_zpl_master_bias_run.c
#     - sph_zpl_master_dark_run.c
#     - sph_zpl_intensity_flat_run.c
#     - sph_quad_image.c
#
Parameter Name:    ESO QC QUAD IMAGE PI ODD NUMBER BADPIXELS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     number of bad pixels
Description:       number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_QUAD_IMAGE_PI_EVEN
#
#    Code references:
#     - sph_zpl_master_bias_run.c
#     - sph_zpl_master_dark_run.c
#     - sph_zpl_intensity_flat_run.c
#     - sph_quad_image.c
#
Parameter Name:    ESO QC QUAD IMAGE PI EVEN NUMBER BADPIXELS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     number of bad pixels
Description:       number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_ZPLEXP_IMG_ODD
```

```
#
#    Code references:
#     - sph_zpl_intensity_flat_imaging_run.c
#
Parameter Name:    ESO QC ZPL EXP IMAGING ODD NUMBER BADPIXELS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     number of bad pixels
Description:       number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_ZPLEXP_IMG_EVEN
#
#    Code references:
#     - sph_zpl_intensity_flat_imaging_run.c
#
Parameter Name:    ESO QC ZPL EXP IMAGING EVEN NUMBER BADPIXELS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     number of bad pixels
Description:       number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_DOUBLE_IMAGE_ODD
#
#    Code references:
#     - sph_zpl_intensity_flat_imaging_run.c
#     - sph_double_image.c
#
Parameter Name:    ESO QC DOUBLE IMAGE ODD(I) NUMBER BADPIXELS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     number of bad pixels
Description:       number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_DOUBLE_IMAGE_EVEN
#
#    Code references:
#     - sph_zpl_intensity_flat_imaging_run.c
#     - sph_double_image.c
#
Parameter Name:    ESO QC DOUBLE IMAGE EVEN(P) NUMBER BADPIXELS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     number of bad pixels
```

```
Description:        number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_LOCI_NRINGS
#
#    Code references:
#     - sph_ird_loci_run.c
#
Parameter Name:     ESO QC LOCI NRINGS
Class:              header|qc-log
Context:            process
Type:               int
Value Format:       %d
Comment Field:      number of rings used by LOCI
Description:        number of rings used by LOCI

#
# Code define: SPH_COMMON_KEYWORD_QC_LOCI_NSUBSECTIONS
#
#    Code references:
#     - sph_ird_loci_run.c
#
Parameter Name:     ESO QC LOCI NSECT
Class:              header|qc-log
Context:            process
Type:               int
Value Format:       %d
Comment Field:      number of sub-sections used by LOCI
Description:        number of sub-sections used by LOCI

#
# Code define: SPH_COMMON_KEYWORD_QC_LOCI_NBADSECTIONS
#
#    Code references:
#     - sph_ird_loci_run.c
#
Parameter Name:     ESO QC LOCI NSECT BAD
Class:              header|qc-log
Context:            process
Type:               int
Value Format:       %d
Comment Field:      number of bad sub-sections
Description:        number of bad sub-sections
```

A.2 IRDIS

```
#
# Code define: SPH_IRD_KEYWORD_SPEC_RES_LINE_X
#
#    Code references:
#     - sph_ird_spectra_resolution_run.c
```

```
#
Parameter Name:    ESO QC SPEC RES POSX LINE
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:
Description:
#
# Code define: SPH_IRD_KEYWORD_SPEC_RES_LINE_Y
#
#    Code references:
#     - sph_ird_spectra_resolution_run.c
#
Parameter Name:    ESO QC SPEC RES POSY LINE
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:
Description:
#
# Code define: SPH_IRD_KEYWORD_SPEC_RES_LINE_WIDTHX
#
#    Code references:
#     - sph_ird_spectra_resolution_run.c
#
Parameter Name:    ESO QC SPEC RES WIDTHX LINE
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:
Description:
#
# Code define: SPH_IRD_KEYWORD_SPEC_RES_LINE_LAMBDA
#
#    Code references:
#     - sph_ird_spectra_resolution_run.c
#
Parameter Name:    ESO QC SPEC RES LAMBDA LINE
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:
Description:
#
# Code define: SPH_IRD_KEYWORD_SPEC_RES_LINE_LAMBDA_WIDTH
#
#    Code references:
#     - sph_ird_spectra_resolution_run.c
#
Parameter Name:    ESO QC SPEC RES LAMBDA WIDTH LINE
```

```
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_NGOODLINES
#
#
Parameter Name:     ESO QC WAVECAL NGOODCOLUMNS
Class:              header|qc-log
Context:            process
Type:               string
Value Format:       %30s
Comment Field:      Number of good pixel columns
Description:        Number of good columns

#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_NBADLINES
#
#
Parameter Name:     ESO QC WAVECAL NBADCOLUMNS
Class:              header|qc-log
Context:            process
Type:               string
Value Format:       %30s
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_NNOFITLINES
#
#
Parameter Name:     ESO QC WAVECAL NNOFITCOLUMNS
Class:              header|qc-log
Context:            process
Type:               string
Value Format:       %30s
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_Y0_MEAN
#
#
Parameter Name:     ESO QC WAVECAL Y0 MEAN
Class:              header|qc-log
Context:            process
Type:               string
Value Format:       %30s
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_Y0_RMS
#
```

```
#
Parameter Name:    ESO QC WAVECAL Y0 RMS
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_C1_MEAN
#
#
Parameter Name:    ESO QC WAVECAL C1 MEAN
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_C1_RMS
#
#
Parameter Name:    ESO QC WAVECAL C1 RMS
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_Y0
#
#
Parameter Name:    ESO QC WAVECAL Y0 COL
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_C1
#
#
Parameter Name:    ESO QC WAVECAL C1 COL
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_CHI
```

```
#
#
Parameter Name:    ESO QC WAVECAL CHI2 COL
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_QC_LAM
#
#
Parameter Name:    ESO QC WAVECAL LAM LINE
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_QC_LAMRMS
#
#
Parameter Name:    ESO QC WAVECAL LAM RMS LINE
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_ESTIM_LINE_POS_LEFT
#
#
Parameter Name:    ESO QC WAVECAL L LINE PIXPOS
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_ESTIM_LINE_POS_RIGHT
#
#
Parameter Name:    ESO QC WAVECAL R LINE PIXPOS
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Comment Field:     Number of good pixel columns
Description:
#
```

```
# Code define: SPH_IRD_KEYWORD_FLUX_LEFT_CORO
#
#    Code references:
#     - sph_ird_flux_calib_run.c
#
Parameter Name:    ESO QC IRD COUNT LEFT CORO ON
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLUX_RIGHT_CORO
#
#    Code references:
#     - sph_ird_flux_calib_run.c
#
Parameter Name:    ESO QC IRD COUNT RIGHT CORO ON
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLUX_LEFT_NO_CORO
#
#    Code references:
#     - sph_ird_flux_calib_run.c
#
Parameter Name:    ESO QC IRD COUNT LEFT CORO OFF
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLUX_RIGHT_NO_CORO
#
#    Code references:
#     - sph_ird_flux_calib_run.c
#
Parameter Name:    ESO QC IRD COUNT RIGHT CORO OFF
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_SPEC_RES_SPATIAL_RES
#
```

```
#    Code references:
#     - sph_ird_spectra_resolution_run.c
#
Parameter Name:    ESO QC AVERAGE SPATIAL LINE FWHM
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_SPEC_RES_LAMBDA_RES
#
#    Code references:
#     - sph_ird_spectra_resolution_run.c
#
Parameter Name:    ESO QC AVERAGE WAVELENGTH LINE FWHM
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_SPEC_RES_RESPOWER
#
#    Code references:
#     - sph_ird_spectra_resolution_run.c
#
Parameter Name:    ESO QC RESOLVING POWER
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_SPEC_RES_RESPOWER_RMS
#
#    Code references:
#     - sph_ird_spectra_resolution_run.c
#
Parameter Name:    ESO QC RESOLVING POWER RMS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_SPEC_RES_NLINES
#
#    Code references:
#     - sph_ird_spectra_resolution_run.c
```

```
#
Parameter Name:    ESO QC NUM SPEC LINES
Class:             header|qc-log
Context:           process
Type:              int
Value Format:      %d
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_DISTMAP_NPOINTS_OBS
#
#    Code references:
#     - sph_ird_distortion_map_run.c
#
Parameter Name:    ESO QC DISTMAP NPOINTS OBS
Class:             header|qc-log
Context:           process
Type:              int
Value Format:      %d
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_DISTMAP_NPOINTS_IN
#
#    Code references:
#     - sph_ird_distortion_map_run.c
#
Parameter Name:    ESO QC DISTMAP NPOINTS IN
Class:             header|qc-log
Context:           process
Type:              int
Value Format:      %d
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_DISTMAP_OPTICAL_AXIS_X
#
#    Code references:
#     - sph_ird_distortion_map_run.c
#
Parameter Name:    ESO QC DISTMAP OPT AXIS X
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_DISTMAP_OPTICAL_AXIS_Y
#
#    Code references:
#     - sph_ird_distortion_map_run.c
#
Parameter Name:    ESO QC DISTMAP OPT AXIS Y
```

```
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLAT_FPN_LEFT
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC FPN LEFT
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLAT_RMS_LEFT
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC RMS LEFT
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLAT_NONLIN_FACTOR_LEFT
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC FLAT NONLINEARITY L
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLAT_MEAN_COUNT_LEFT
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC FLAT MEAN LEFT
Class:              header|qc-log
Context:            process
```

```
Type:               double
Value Format:       %.2f
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLAT_FPN_RIGHT
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC FPN RIGHT
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLAT_RMS_RIGHT
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC RMS RIGHT
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLAT_NONLIN_FACTOR_RIGHT
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC FLAT NONLINEARITY R
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLAT_MEAN_COUNT_RIGHT
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC FLAT MEAN RIGHT
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
```

```
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLAT_LAMP_FLUX_LEFT
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC LAMP FLUX AVG LEFT
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLAT_LAMP_FLUX_RIGHT
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC LAMP FLUX AVG RIGHT
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLAT_LAMP_FLUX_STDEV_LEFT
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC LAMP FLUX STDEV LEFT
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Number of good pixel columns
Description:
#
# Code define: SPH_IRD_KEYWORD_FLAT_LAMP_FLUX_STDEV_RIGHT
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:     ESO QC LAMP FLUX STDEV RIGHT
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Number of good pixel columns
Description:
```

A.3 IFS

```
#
# Code define: SPH_IFS_KEYWORD_SPECPOS_QC_NREGS
#
#    Code references:
#     - sph_ifs_spectra_positions_run.c
#
Parameter Name:    ESO QC NUMBER SPECTRA
Class:             header|qc-log
Context:           process
Type:              int
Value Format:      %d
Comment Field:     Number of spectra found
Description:       Number of spectra found

#
# Code define: SPH_IFS_KEYWORD_SPECPOS_QC_THRESHOLD
#
#    Code references:
#     - sph_ifs_spectra_positions_run.c
#
Parameter Name:    ESO QC THRESHOLD USED
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Threshold used for spectra detection
Description:       Threshold used for spectra detection

#
# Code define: SPH_IFS_KEYWORD_SPECPOS_QC_SCALE
#
#    Code references:
#     - sph_ifs_spectra_positions_run.c
#
Parameter Name:    ESO QC SCALE MEASURED
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Comment Field:     Scale of specpos model as measured
Description:       Scale of specpos model as measured

#
# Code define: SPH_IFS_KEYWORD_CAL_BG_QC_MEAN
#
#    Code references:
#     - sph_ifs_cal_background_run.c
#
Parameter Name:    ESO QC BACKGROUND MEAN
Class:             header|qc-log
Context:           process
```

```
Type:               double
Value Format:       %.2f
Comment Field:      beckground mean value
Description:        Background mean value

#
# Code define: SPH_IFS_KEYWORD_CAL_BG_QC_RMS
#
#    Code references:
#     - sph_ifs_cal_background_run.c
#
Parameter Name:     ESO QC BACKGROUND RMS
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Background RMS
Description:        Bckground RMS

#
# Code define: SPH_IFS_KEYWORD_CAL_BG_QC_MEDIAN
#
#    Code references:
#     - sph_ifs_cal_background_run.c
#
Parameter Name:     ESO QC BACKGROUND MEDIAN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Comment Field:      Background median value
Description:        Background median value

#
# Code define: SPH_IFS_KEYWORD_PREAMPCORR_MEAN
#
#
Parameter Name:     ESO QC PREAMP CORR MEAN
Class:              header|qc-log
Context:            process
Type:               string
Value Format:       %30s
Comment Field:      Mean value of preamp correlation
Description:        Mean value of preamp correlation

#
# Code define: SPH_IFS_KEYWORD_PREAMPCORR_MEDIAN
#
#
Parameter Name:     ESO QC PREAMP CORR MEDIAN
Class:              header|qc-log
Context:            process
Type:               string
Value Format:       %30s
```

```
Comment Field:      Median value of preamp correlation
Description:        Median value of preamp correlation

#
# Code define: SPH_IFS_KEYWORD_PREAMPCORR_RMS
#
#
Parameter Name:     ESO QC PREAMP CORR RMS
Class:              header|qc-log
Context:            process
Type:               string
Value Format:       %30s
Comment Field:      RMS of preamp correlation
Description:        RMS of preamp correlation

#
# Code define: SPH_IFS_KEYWORD_DISTMAP_NPOINTS_OBS
#
#    Code references:
#     - sph_ifs_distortion_map_run.c
#
Parameter Name:     ESO QC DISTMAP NPOINTS OBS
Class:              header|qc-log
Context:            process
Type:               int
Value Format:       %d
Comment Field:      Number of points observed for distortion map
Description:        Number of points observed for distortion map
```

A.4 ZIMPOL

There are currently no specific ZIMPOL QC keywords. See common.