	REF: **VLT-TRE-SPH-14690-660/1/0** Issue: 1 Version 47 Date: 2023-05-10 Pages: 220

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-lookassessment of data, in the generation of master calibration data, in thereduction 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 reductionsequence with the SPHERE pipeline.This manual is a complete description of the data reduction recipes offeredby the SPHERE pipeline, reflecting the status of the SPHERE pipeline asof 29th July 2016 (version 0.19.0).

1.2 Scope of this document

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

The main purpose of this document is to present and explain the data reductionsoftware (in form of a library) for SPHERE in general and IFS, IRDIS andZIMPOL in particular.The structure and content of this document follows the guidelines set outin 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 4.5.1of the “Data Flow for VLT/VLTI Instruments Deliverables Specifications”closely with the exception of an added introduction (chapter 1 in thisdocument) and an overview (chapter 2).

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

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

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
1	1		Restructured	02-11-2015	V 16
1	16	3.2,3.3,3.4	Updated Workflows and recipe descriptions	15-02-2016	
1	19	3.2,3.3,3.4	Updated Workflows and recipe descriptions	25-07-2016	

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 datareduction 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 informationnecessary to monitor instrument performance.
- Master calibration product creation: pipelines are used to produce mastercalibration 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 availablemaster calibration products and by the algorithmic implementation of thepipelines themselves.In particular, adopted automatic reduction strategies may not be suitableor optimal for all scientific goals.

Instrument pipelines consist of a set of data processing modules that canbe called from the command line, from the automatic data management toolsavailable 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 3.1of this manual.Workflows for individual data reduction cascades and detailed descriptionsof each recipe's inputs, outputs, parameters and operations can be foundin Sections 3.2, 3.3, and 3.4.

The SPHERE instrument and the different types of SPHERE raw frames and auxiliarydata are described in Chapter 4.

A detailed mathematical description of operations carried out inside recipesis given in Chapter 5.

In Chapter 6the installation of the SPHERE pipeline recipes is described together witha few solutions to frequent problems.

2.1 The SPHERE Instrument: IFS, IRDIS and ZIMPOL

In addition to this document, there is the SPHERE user manual [RD10] whichgives a brief indtroduction to the instrument, as well as a general descriptionof the available observing instruments and modes, and details about settingup actual observations.

Chapter 3

Pipeline Guide

This section describes the most immediate usage of the SPHERE pipeline recipes.If you are unfamiliar with ESO pipelines and the plugins called recipes,you can find a basic description of how to operate thse things in 3.1.An example workflow for a data set supposedly taken in an IRDIS

observing sequence is described in 3.1.4.More detailed information on the workflow for each instrument and individualrecipes can be found in ...

3.1 Quick Start

3.1.1 An introduction to Gasgano and EsoRex

Before being able to call pipeline recipes on a set of data, the data mustbe opportunely classified, and associated with the appropriate calibrations.The Data Classification consists of tasks such as: "What kind of data amI?", e.g., DARK, "to which group do I belong?", e.g., to a particular ObservationBlock or template.Data Association is the process of selecting appropriate calibration datafor the reduction of a set of raw science frames.Typically, a set of frames can be associated if they share a number ofproperties, such as instrument and detector configuration.As all the required information is stored in the FITS headers, data associationis based on a set of keywords (called "association keywords") and is specificto 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 dataclassification.An instrument pipeline consists of a set of data processing modules thatcan be called from different host applications, either from the commandline with Esorex, from the automatic data management tools available atParanal, or from the graphical Gasgano tool.Gasgano is a data management tool that simplifies the data organisationprocess, offering automatic data classification and making the data associationeasier (even if automatic association of frames is not yet provided).

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

3.1.2 Using Gasgano

To get familiar with the SPHERE pipeline recipes and their usage, it isadvisable to begin with Gasgano, because it provides a complete graphicinterface for data browsing, classification and asso-

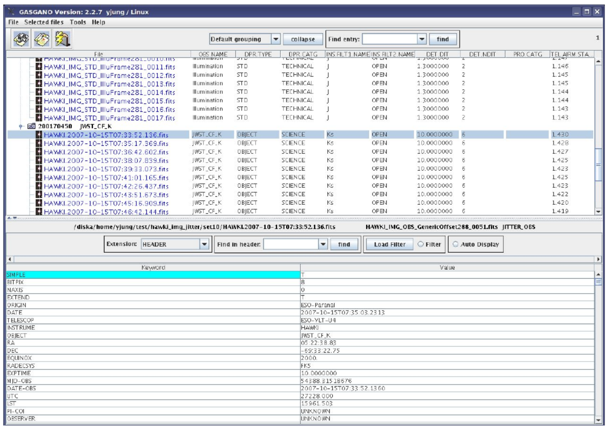

Figure 3.1: The Gasgano main window

ciation, and offersseveral other utilities such as easy access to recipes documentation andpreferred data display tools.Gasgano can be started from the Command Line Interface in the followingway: gasgano & Figure 4.2.1 shows the Gasgano main window.

With the pull-down-menu File->Add/Remove Files directories containing SPHEREdata can be added.The data are hierarchically organised as preferred by the user.After each file name are shown the classification, the template id, theoriginal file name, the template exposure number and the number of exposuresin the template.More information about a single frame can be obtained by clicking on itsname: the corresponding FITS file header will be displayed on the bottompanel, where specific keywords can be opportunely filtered and searched.Images and tables may be easily displayed using the viewers specified inthe appropriate Preferences fields.

Frames can be selected from the main window with a <CTRL>-left-click forprocessing by the appropriate recipe: on Figure 4.2 a set of calibrationFITS-files have been selected and after selecting the appropriate recipe,the depicted Gasgano recipe execution window will open, having all thespecified 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 Parameterspanel (on top).The window contents might be saved for later use by selecting the SaveCurrent 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 panelat bottom, and in case of successful completion the products will be listedon the Output Frames panel, where they can be easily viewed and locatedback on the Gasgano main window.Please refer to the Gasgano User's Manual [RD9] for a more complete descriptionof the Gasgano interface.

3.1.3 Using EsoRex

EsoRex is a command line utility for running pipeline recipes.It may be embedded by users into data reduction scripts for the automationof processing tasks.On the other side, EsoRex doesn't offer all the facilities available withGasgano, 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 andthe 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 DOcategory in an ASCII file, the set-of-frames (SOF), that is required whenlaunching a recipe.Here is an example of SOF, valid for the sph_ird_instrument_flat recipe:

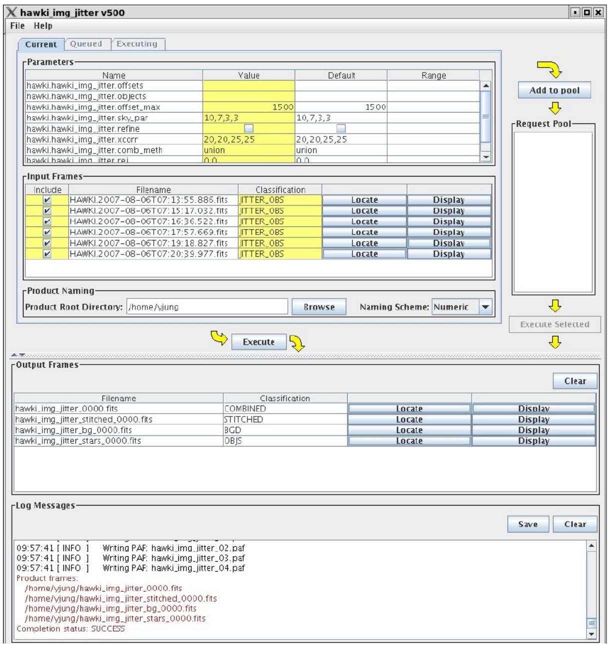

Figure 3.2: The Gasgano recipe execution window

```
/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 theDO category specified by the user in the SOF.The reason of this lack of control is that SPHERE recipes are just onecomponent of the complete pipeline running on Paranal, where the task ofdata classification and association is carried out by separate applications.Using Gasgano as an interface to the pipeline recipes will however ensurea correct classification of all the data frames, assigning the appropriateDO category to each one of them (see section 4.2.1).A recipe handling an incorrect SOF may stop or display unclear error messagesat 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]) runthe command:

```
esorex --help
```

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

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

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

```
esorex --recipes
```

All recipe parameters (aliases) and their default values can be displayedby 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 shellprompt:

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

3.1.3.1 Recipe configuration:

Each pipeline recipe may be assigned an EsoRex configuration file, containingthe 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 thefile 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 (controllingthe 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 linesare 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 ina cascade of data reduction steps, each controlled by its own parameters.For this reason and to prevent parameter name clashes we specify as parameterprefix not only the instrument name but also the name of the step theyrefer 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 specifiedon 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 recipebut, 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 specifiedwhen launching a recipe.

3.1.3.2 Recipe execution:

A recipe can be run by specifying its name to EsoRex, together with thename of a set-of frames.For instance, the following command line would be used to run the recipesph_ifs_master_dark for processing the files specified in the set-of-framessph_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 usedconfiguration file, or by specifying new parameter values on the commandline using the command line options defined for this purpose.Such command line options should be inserted after the recipe name andbefore the SOF name, and they will supersede the system defaults and/orthe configuration file settings.For instance, to set the sph_ifs_master_dark reject_high parameter to 4the 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].

3.1.4 Example IRDIS Reduction

3.1.4.1 Data set

The data set used in this example is the one used to generate the test sequenceIRDIS-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 andoutlined below.There is no subdirectory structure in the tar file, so it is best to createa suitable subdirectory from your working directory and unpack the tarfile in there.the following examples will assume that this directiry is named "Raw".

3.1.4.2 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 separatefile.View it with your favourite FITS viewer and compare with the input filesto verify that it is indeed the mean of the inputs.Notice that the master_dark.fits file has 3 additional extensions – lookat 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 detailand what is stored in the other extensions.

3.1.4.3 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 (bad pixel) 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.

3.1.4.4 Distortion Map

To correct for the instruments dostortion, a specific map is generated fromdedicated 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 thresholdingand peak finding, this one is quite sensitive to the threshold providedand a look at the data may be required before calling the recipe.

Note that since no reference grid is supplied, the recipe assumes that thedata 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 theresulting file distortion_map.fits, denoting the distortion in x and y ateach pixel.

3.1.4.5 Star center table creation

Now before actually doing the science reduction it is currently necessaryto run a prototype version of the sph_ird_star_center recipe.This recipe is responsible to create a table of field centers, which arecrucial 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
waffle_lowmaskIRD_STATIC_BADPIXELMAP.fits        IRD_STATIC_WAFFLEMAP
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 product called star_center.fits that contains a fitstable 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 correspondingparameters passed to the recipe are quite sensitive, and experimentingis always welcome!

Note also that the waffle_lowmaskIRD_STATIC_BADPIXELMAP.fits file is not actually raw data,but simply a mask provided along with the pipeline! You are free to adapt thismask to improve results...

3.1.4.6 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 identicalto 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 directoryare included.This is simply to make sure you dont have to wait for several hours forall 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 andlearn 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 betweenthe two channels...

3.2 IRDIS

IRDIS offers the following basic observing modes:

- Dual-Band Imaging (DBI)
- Classical Imaging (CI)
- Long-Slit Spectroscopy (LSS)
- Dual-Band Polarimetry (DPI)

The calibration cascade consists of a number of basic calibrations whichapply to all modes.The recipes for these basic calibrations are presented in .Mode specific calibration recipes are presented in the subsequent subsections.Each recipe description lists the recipe inputs, both raw frames and calibration products.Also given is a description of the recipe's parameters, and of the outputproduct.From these descriptions it should be possible to construct a workflow.A summary is additionally given in .

3.2.1 Basic IRDIS Calibrations

3.2.1.1 sph_ird_gain

3.2.1.1.1 Purpose:

Measure the detector gain

3.2.1.1.2 Input frames:

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

3.2.1.1.3 Raw frame keywords used:

none

3.2.1.1.4 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.save_ addprod	bool	Controls if additional products, in this case a badpixel map and the nonlinearity map should be saved. Note that these will only be created in the first place if The vacca method is not used (see below) AND the fitting order is greater than 1	0	-
ird.gain.nonlin_ filename	string	The output filename for the non-linearity map. Please also see the esorex documentation	ird_ nonlin_ map.fits	-
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	-

3.2.1.1.5 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 should be different either by increasing the intensity of the illumination source or by using different exposure times. 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, which corrects 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 algorithms to calculate the gain, one straightforward fitting algorithm and a more complex fitting algorithm that takes the correct number of fowler samples into account. The second algorithm is switched on using the vacca user parameter and is preferable 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 when an 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 all input 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 variance image has been determined the median of the mean image and the corresponding variance is taken as one data point. The collection of input cubes then lead to a collection of data points of median and variance, giving measurements of the variance vs. median relation for the detector. This is then fitted using a polynomial of the specified order (usually 1 or 2). The slope of this curve is the inverse of the gain while the offset gives an estimate of the read out noise. Note that the read out noise estimate 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 ron are 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 noise properties 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 product FITS file. Note that the pixel-by-pixel gain values are often very noisy and can not be used to obtain precise 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 (quadratic) coefficient of the individual pixel fits is saved in an additional FITS file. All pixels that have second order quadratic 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.

3.2.1.1.6 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.

Name	Type	Description
IRD_ NONLIN_ BADPIX	FITS[Im(1)]	A simple image flagging all non linear pixels.

3.2.1.2 sph_ird_master_dark

3.2.1.2.1 Purpose:

Creation of the master dark frame

3.2.1.2.2 Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ DARK_ RAW	Raw data	No	1	Any

3.2.1.2.3 Raw frame keywords used:

none

3.2.1.2.4 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	-
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	-

3.2.1.2.5 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 collapse algorithm specified (usually the clean_mean algorithm). After all input frames are combined in this way, the badpixels are determined on 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 the specified 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 separate hotpixel map is also written out.

3.2.1.2.6 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.

3.2.1.3 sph_ird_ins_bg

3.2.1.3.1 Purpose:

Creation of an instrument background

3.2.1.3.2 Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ INS_ BG_ RAW	Raw data	No	1	Any
IRD_ INSTRUMENT_ MODEL	Calibration	Yes	0	1
IRD_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1

3.2.1.3.3 Raw frame keywords used:

none

3.2.1.3.4 Parameters:

Name	Type	Description	Default	Allowed vals.
ird.ins_ bg.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	ins_ bg.fits	-
ird.ins_ bg.save_ addprod	bool	Flag to signal whether additional products - in this case the smoothed background - should be saved.	0	-

Name	Type	Description	Default	Allowed vals.
ird.ins_ bg.lsf_ outfilename	string	Flag to signal whether additional products - in this case the smoothed background - should be saved.	ins_ bg_ fit.fits	-
ird.ins_ bg.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.ins_ bg.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	0	0-20
ird.ins_ bg.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	0	0-20
ird.ins_ bg.fitorder	int	The fitting order to use for the 2D polynomial fit of the background. smoothing range double	2	1-7

3.2.1.3.5 Description:

This recipe deals with the creation of the instrument background calibration frame. Only raw frames are used in this recipe. The background is created by combining the input raw frames using the collapse algorithm specified (usually the clean_mean algorithm). Contrary to the master_dark recipe, no badpixel maps are created by this recipe, since thresholding is much more difficult on the background frames. The recommended way to generate a reliable badpixel map is to use the master_dark recipe on suitable dark frames! MASTER_DARK, INS_BG, and SKY_BG frames are all for subtraction from raw input frames, thus they should never be corrected for flat fields or other dark/background frames!

3.2.1.3.6 Products:

Name	Type	Description
IRD_ INS_ BG	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_ INS_ 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. Left and right frame fits are obtained separately, but inserted into a full frame result!

3.2.1.4 sph_ird_sky_bg

3.2.1.4.1 Purpose:

Creation of a sky background

3.2.1.4.2 Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ SKY_ BG_ RAW	Raw data	No	1	Any
IRD_ INSTRUMENT_ MODEL	Calibration	Yes	0	1
IRD_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1

3.2.1.4.3 Raw frame keywords used:

none

3.2.1.4.4 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.save_ addprod	bool	Flag to signal whether additional products - in this case the smoothed background - should be saved.	0	-
ird.sky_ bg.lsf_ outfilename	string	Flag to signal whether additional products - in this case the smoothed background - should be saved.	sky_ bg_ fit.fits	-
ird.sky_ bg.coll_ alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = Clean mean.	2	0,1,2
ird.sky_ bg.clean_ mean.reject_ low	int	The clean mean reject pixels on low end.	0	0-20
ird.sky_ bg.clean_ mean.reject_ high	int	The clean mean reject pixels on high end.	0	0-20
ird.sky_ bg.fitorder	int	The fitting order to use for the 2D polynomial fit of the background. smoothing range double	2	1-7

3.2.1.4.5 Description:

This recipe deals with the creation of the sky background calibration frame. Only raw frames are used in this recipe. The background is created by combining the input raw frames using the collapse algorithm specified (usually the clean_mean algorithm). Contrary to the master_dark recipe, no badpixel maps are created by this recipe, since thresholding is much more difficult on the background frames. The recommended way to generate a reliable badpixel map is to use the master_dark recipe on suitable dark frames! MASTER_DARK, INS_BG, and SKY_BG frames are all for subtraction from raw input frames, thus they are never corrected for flat fields or other dark/background frames!

3.2.1.4.6 Products:

Name	Type	Description

Name	Type	Description
IRD_ SKY_ BG	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_ 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. Left and right frame fits are obtained separately, but inserted into a full frame result!

3.2.1.5 sph_ ird_ instrument_ flat

3.2.1.5.1 Purpose:

Determine the instrument flat field

3.2.1.5.2 Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ FLAT_ FIELD_ RAW	Raw data	No	1	500
IRD_ MASTER_ DARK	Calibration	Yes	0	1
IRD_ INS_ BG	Calibration	Yes	0	1
IRD_ INS_ BG_ FIT	Calibration	Yes	0	1
IRD_ DARK_ RAW	Raw data	Yes	0	500
IRD_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1
IRD_ INSTRUMENT_ MODEL	Calibration	Yes	0	1

3.2.1.5.3 Raw frame keywords used:

none

3.2.1.5.4 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	-

Name	Type	Description	Default	Allowed vals.
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
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	-

3.2.1.5.5 Description:

The instrument flat field recipe for IRDIS is very similar as the detector flat field 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 that input frames are consistently for one particular instrument configuration and that the 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. Dark handling in this recipe is special: For best results, the recipe needs a series of raw dark frames with DITs matching those of the raw flat field frames. If raw darks are provided, the recipe will for each raw flat field frame select the raw dark with the closest matching DIT and subtract it. the recipe will not insist on perfectly matching DITs, this is the user's responsibility! If raw darks are unavailble (e.g. in a automated pipeline context), the recipe can also apply standard irdis background frames. Note that in this case a single dark frame will be subtracted from all input frames. Best results can thus be achieved only if all input raw flat fields have the same DIT (matching that of the background, of course) and lamp intensity is varied to achieve variable flux. The order of selecting what actually happens is the following:

1. If raw darks are available, all others are ignored.
2. Else if an INS_BG_FIT is available, this one is chosen.
3. Else if an INS_BG is available, this one is chosen.
4. Else if a MASTER_DARK is available, this one is chosen.

The recipe will also run without any dark at all! It is thus the user's responsibility to supply adequate background frames to achieve the best possible results! The recipe creates the flats as follows: All raw frames are read in and dark subtracted. The dark subtraction is performed differently than for other recipes, and rather than master darks, the recipe actually uses raw dark or background frames. Since the background varies significantly depending on the chosen detector integration time, a dark with a matching exposure time needs to be subtracted for each flat. If a specific irdis instrument model is 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 right detector windows. In the next step, a mask of the illuminated region is created by combining all input exposures and using a thresholding above the given input threshold value to identify illuminated regions and 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 then applied to the left and right windows **seperately**.

1. The mean value is determined for the respective window for all 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 method or 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) can be 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 user parameters 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 above a given threshold value. Note that non-linearity pixel determination is performed on 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 badpixels that are stored in the master flat field itself as produced by this recipe (the second extension of the main recipe product) contain all the badpixels at this point in the cascade. Pixels that were 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 the flat field recipe itself deemed to be bad. This does not necessarily include all the badpixels from the static badpixel input file.

3.2.1.5.6 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
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.

3.2.1.6 sph_ird_distortion_map

3.2.1.6.1 Purpose:

Creation of the total distortion map

3.2.1.6.2 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_ INS_ BG	Calibration	Yes	0	1
IRD_ INS_ BG_ FIT	Calibration	Yes	0	1
IRD_ SKY_ BG	Calibration	Yes	0	1
IRD_ SKY_ BG_ FIT	Calibration	Yes	0	1
IRD_ FLAT_ FIELD	Calibration	Yes	0	1
IRD_ POINT_ PATTERN	Calibration	Yes	0	1

3.2.1.6.3 Raw frame keywords used:

none

3.2.1.6.4 Parameters:

Name	Type	Description	Default	Allowed vals.
ird.distortion_ map.convert	bool	If this flag is set to TRUE (FALSE is default), the recipe will not create a distortion map. Instead it will convert the point pattern table which must be provided in the sof to either FITS or ASCII format. The filename specified with the outfilename parameter is used for the output.	0	-
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 degree of the 2D-polynomial fitted to the distortion. The degree is used for both the X- and the Y-direction.	3	0-8
ird.distortion_ map.max_ distortion	double	The maximal distortion to correct for [pixel]. Any observed point that is found to be further than this threshold from its matching calibration point is excluded from the fitting procedure. To avoid an incorrect matching between an observed point and its calibration point this threshold should not be too large. For a grid of equidistant calibration points this upper limit is half the distance between two neigboring calibration points. For a calibration mask with 73 by 73 points on a 1k by 1k detector this limit is just over 7 pixels.	7.0	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	-

Name	Type	Description	Default	Allowed vals.
ird.distortion_ map.user_ cent	bool	the recipe uses as optical centre the coordinate of the point that is closest to the detector centre of the point pattern, while for the right channel the same point is used.	0	-
ird.distortion_ map.cent_ left_ x	double	The optical centre of the left FOV. This is only used if the user_cent parameter is set to TRUE.	512.5	-
ird.distortion_ map.cent_ left_ y	double	The optical centre of the left FOV. This is only used if the user_cent parameter is set to TRUE.	512.5	-
ird.distortion_ map.cent_ right_ x	double	The optical centre of the right FOV. This is only used if the user_cent parameter is set to TRUE.	512.5	-
ird.distortion_ map.cent_ right_ y	double	The optical centre of the right FOV. This is only used if the user_cent parameter is set to TRUE.	512.5	-
ird.distortion_ map.align_ right	bool	When set to true, the distortion correction of the right channel has a fixed shift added to it to align it with the left channel. The added shift is the difference between the optical axis of the left and right channel.	0	-

3.2.1.6.5 Description:

This recipe creates a map of the distortion for the instrument. The raw frames are first reduced like standard science frames in field stabilised mode without dithering. The frame combination is simply done using a clean mean, mean or median combination. If given as input, a dark is subtracted and a flat field applied. Dark handling follows the usual strategy, see man page of sph_ird_science_dbi. For this recipe, providing a dark is optional - it will happily look for points also without subtracting anything in advance! 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 input point 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 optical axis. This is assumed to be the coordinates of the point closest to the geometrical centre of the point pattern. The geometrical centre of the point pattern is calculated averaging the positions of all the points belonging to the point pattern. The centre values are measured relative to the extension of the product. In order to obtain the pixel coordinates in the raw frame 1024 has to be added to the right channel x coordinate.
3. shifting the input point pattern so that its most central point has the same coordinates as the optical axis. This means that the central points on real and expected point pattern fall exactly on top 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 distortion value given as parameter to the recipe are removed.

6. The resulting distortion measurements are then used to calculate two 2D-polynomial fits to create a distortion map in both X and Y for all pixels. Each polynomial has its center of origin on the detector center.

The main product of the recipe is a multi-extension file that gives the distortion map for each IRDIS field of view separately. Other recipes use the polynomial fit as stored in the header of extension 0 and 8 to apply the distortion map. 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 distortion map 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 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 by feeding the full detector control image back into a second run of the distortion map recipe. The resulting distortion map then gives the distortion residuals – and these should all be close to 0. The polynomial fit is available as QC parameters in the distortion map. The polynomial fitting is performed on the point patterns shifted by the centre of the optical axis. The fitting generates two polynomials: $p_x(x, y)$ and $p_y(x, y)$ where the first provides the shift in the x direction for a point of coordinates (x, y). $p_y(x, y)$ refers to the shift in the y coordinates. The coefficients of the polynomials are written as QC parameters in the form "ESO DRS DIST L X COEFF i_j" where "L" indicates that the QC parameter belongs to the left FOV (for right FOV "R" is used). The letter "X" indicates that the coefficient belongs to $p_x(x, y)$ ("Y" is used for $p_y(x, y)$). i and j are the powers of x and y, i.e. $x^i y^j$. The pin point static calibration is shifted to the closest pin-point image. This shift is stored as the estimated optical axis in the keywords "ESO QC DISTMAP OPT AXIS X" and "ESO QC DISTMAP OPT AXIS Y". Then the polynomial fit is applied as described above and saved as DISTORTION MAP, which is used in later steps of the cascade to correct the optical distortion.

3.2.1.6.6 Products:

Name	Type	Description

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.

3.2.1.7 sph_ird_star_center

3.2.1.7.1 Purpose:

Determine the field centre

3.2.1.7.2 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_ INS_ BG	Calibration	Yes	0	1
IRD_ INS_ BG_ FIT	Calibration	Yes	0	1
IRD_ SKY_ BG	Calibration	Yes	0	1
IRD_ SKY_ BG_ FIT	Calibration	Yes	0	1
IRD_ FLAT_ FIELD	Calibration	Yes	0	1
IRD_ STATIC_ WAFFLEMAP	Calibration	Yes	0	1

3.2.1.7.3 Raw frame keywords used:

none

3.2.1.7.4 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

Name	Type	Description	Default	Allowed vals.
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	-
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.save_ interprod	bool	Flag to signal if intermediate products should be kept	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	-

3.2.1.7.5 Description:

This recipe creates a table with centre star positions. The input raw frames are each reduced by subtracting the dark and applying the flat provided. Dark handling follows the usual strategy, see man page of sph_ird_science_dbi. For this recipe, providing a dark is optional - it will happily look for points also without subtracting anything in advance! 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 dimensions 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 extracted and left and right part are seperately analysed using a aperture detection algorithm. The aperture detection algorithm detects all connected regions of 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 coronograph is out and no waffle stars are formed). After frame centers have been determined for all waffle images an internal table is created with an entry for each waffle image, giving the time of the start 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.

3.2.1.7.6 Known Issues:

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

3.2.1.7.7 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.

3.2.2 IRDIS Dual-Band Imaging (DBI) Specific Calibrations and Science

3.2.2.1 sph_ird_flux_calib

3.2.2.1.1 Purpose:

Calibrate the effect of coronagraph

3.2.2.1.2 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

3.2.2.1.3 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.

3.2.2.1.4 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

3.2.2.1.5 Description:

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

3.2.2.1.6 Products:

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

3.2.2.2 sph_ird_science_dbi

3.2.2.2.1 Purpose:

Science calibration, DBI mode.

3.2.2.2.2 Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_SCIENCE_DBI_RAW	Raw data	No	1	500
IRD_MASTER_DARK	Calibration	Yes	0	1
IRD_INS_BG	Calibration	Yes	0	1
IRD_INS_BG_FIT	Calibration	Yes	0	1
IRD_SKY_BG	Calibration	Yes	0	1
IRD_SKY_BG_FIT	Calibration	Yes	0	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

3.2.2.2.3 Raw frame keywords used:

Keyword	Type	Optional	Description
ESO INS1 OPTI2 NAME	string	No	

3.2.2.2.4 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 required, products for the left and right fields	1	-
ird.science_dbi.save_interprod	bool	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!	1	-
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.outfilename_right	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	science_dbi_right.fits	-

Name	Type	Description	Default	Allowed vals.
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
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 filter, 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 frame combination. 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	The star radius [arcsecond] used for the Strehl ratio estimate. A negative value disables the estimation. This option is ignored in absence of a filter table frame.	2.0	-

Name	Type	Description	Default	Allowed vals.
ird.science_ dbi.bg_ r1	double	The internal radius [arcsecond] of the background used for the Strehl ratio estimate. This option is ignored in absence of a filter table frame.	2.0	-
ird.science_ dbi.bg_ r2	double	The external radius [arcsecond] of the background used for the Strehl ratio estimate. This option is ignored in absence of a filter table frame.	3.0	-

3.2.2.2.5 Description:

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

1. The input raw frames are dark subtracted, see below
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 ***(does not work)***
4. the left and right IRDIS subframes are extracted using the IRDIS instrument model as 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 ***(switched off by default)***. If the filter radius f_r is set to a value larger than 0, a top hat 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 are good 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 the field 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 the field center table provided in the input of the recipe.

If the angles are not specified by the field center table, and ADI is turned on, the images will be aligned by rotating them all to a parallactic angle of zero. Then these frames are combined using the selected combination algorithm.

- if a weighted mean is selected, a weightmap is calculated first taking the median frame as a reference frame and weighing down the other frames depending on the difference in values. Note that this is still a very experimental option and it is still to be defined what weighting sheme would be optimal. All badpixels get assigned the weight of 0. Frames are then combined 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 procedure ignores 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, following the layout for a double master frame: the first four extensions being the image, badpixelmap, N map, and rms for the left field and the 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 in the 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. In case a number of $n < 8$ neighbour pixels are also bad, 8 - n values are used. 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.
- For a point-source an estimate of the Strehl ratio may be useful. The presence of a filter table frame will enable the estimation, which on failure will do nothing and on success will insert Strehl related QC parameters into each product header.

Dark handling: This recipe will not run without a supplied dark or background frame. Possible frames to be subtracted are SKY_BG_FIT, SKY_BG, INS_BG_FIT, INS_BG, and MASTER_DARK. DIT and readout mode should match the science data, but this is not verified by the recipe! For everything except MASTER_DARK, it is a wise idea to also match the filter configuration! If you provide more than one of the optional frames, a choice will be made according to the following priorization:

1. If a SKY_BG_FIT is available, this one is chosen.
2. Else if a SKY_BG is available, this one is chosen.
3. Else if an INS_BG_FIT is available, this one is chosen.
4. Else if an INS_BG is available, this one is chosen.
5. Else if a MASTER_DARK is available, this one is chosen.
6. Else an error will be thrown and execution terminated.

3.2.2.2.6 Products:

Name	Type	Description

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 additional products.

3.2.3 IRDIS Classical Imaging (CI) Specific Calibrations and Science

3.2.3.1 sph_ ird_ science_ imaging

3.2.3.1.1 Purpose:

Science calibration, imaging mode.

3.2.3.1.2 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_ INS_ BG	Calibration	Yes	0	1
IRD_ INS_ BG_ FIT	Calibration	Yes	0	1
IRD_ SKY_ BG	Calibration	Yes	0	1
IRD_ SKY_ BG_ FIT	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_ FILTER_ TABLE	Calibration	Yes	0	1
IRD_ PHOT_ STAR_ TABLE	Calibration	Yes	0	1

3.2.3.1.3 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.

3.2.3.1.4 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	-

Name	Type	Description	Default	Allowed vals.
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_ imaging.keep_ fctable	bool	if set to TRUE the recipes internall created field center tables are not deleted.	0	-
ird.science_ imaging.save_ addprod	bool	if set to TRUE the recipe will save additional products (left and write fields, in this case!)	0	-
ird.science_ imaging.use_ adi	bool	Flag to control usage of 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
ird.science_ imaging.filter_ method	int	FFT filter method to use. 0 is none, 1 is top hat filter, 2 is Fermi filter, 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 frame combination. 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
ird.science_ imaging.star_ r	double	The star radius [arcsecond] used for the Strehl ratio estimate. A negative value disables the estimation. When AO is enabled and 0 (default) is provided 2 arcseconds are used. When AO is disabled and 0 is provided a radius corresponding to 277 PIXEL is used. This option is ignored in absence of a IRD_FILTER_TABLE frame.	0.0	-

Name	Type	Description	Default	Allowed vals.
ird.science_ imaging.bg_ r1	double	The internal radius [arcsecond] of the background used for the Strehl ratio estimate. When AO is enabled and 0 (default) is provided 2 arcseconds are used. When AO is disabled and 0 is provided a radius corresponding to 277 PIXEL is used. This option is ignored in absence of a IRD_FILTER_TABLE frame.	0.0	-
ird.science_ imaging.bg_ r2	double	The external radius [arcsecond] of the background used for the Strehl ratio estimate. When AO is enabled and 0 (default) is provided 3 arcseconds are used. When AO is disabled and 0 is provided a radius corresponding to all the PIXELS in the image is used. This option is ignored in absence of a IRD_FILTER_TABLE frame.	0.0	-

3.2.3.1.5 Description:

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

1. The input raw frames are dark subtracted, see below
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 ***(does not work)***
4. the left and right IRDIS subframes are extracted using the IRDIS instrument model as 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 ***(switched off by default)***. If the filter radius f_r is set to a value larger than 0, a top hat 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 are good 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 the selected combination algorithm.

- if a weighted mean is selected, a weightmap is calculated first taking the median frame as a reference frame and weighing down the other frames depending on the difference in values. Note that this is still a very experimental option and it is still to be defined what weighting sheme would be optimal. All badpixels get assigned the weight of 0. Frames are then combined 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 procedure ignores 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, following the layout for a double master frame: the first four extensions being the image, badpixelmap, N map, and rms for the left field and the second set of four extensions being the equivalent for the right field. Some additional notes:

- For a point-source an estimate of the Strehl ratio may be useful. The presence of a filter table frame will enable the estimation, which on failure will do nothing and on success will insert Strehl related QC parameters into each product header.
- If additionally a standard star table is supplied and the target observed can be found in that table, an estimate of the zeropoint is also computed.

Dark handling: This recipe will not run without a supplied dark or background frame. Possible frames to be subtracted are SKY_BG_FIT, SKY_BG, INS_BG_FIT, INS_BG, and MASTER_DARK. DIT and readout mode should match the science data, but this is not verified by the recipe! For everything except MASTER_DARK, it is a wise idea to also match the filter configuration! Strehl Ratio Calculation: The recipe calculates the Strehl Ratio following these steps:

1. Optionally correct the residual local sky background evaluated in an annular region centered on the expected peak of the Point Spread Function (PSF). The region extendes between bg_r1 and bg_r2. The center is the centroid of the apertures found in the image.
2. The PSF is identified and its integrated flux is normalized to 1. The flux is integrated in a circular region having radius star_r;
3. The PSF barycentre is computed and used to generate the theoretical normalised PSF. It depends on the pixel scale (PIXSCAL extracted from the raw header) and on the nomial filter wavelength extracted from SPH_IRD_TAG_FILTER_TABLE_CALIB;
4. The Strehl Ratio is the ratio between the maximum intensities of the PSF and the theoretical PSF.

Zeropoint Calculation: The recipe calculates the Zeropoint following these steps:

1. The zeropoint calculation uses the integrated flux (f) of the star determined as part of the Strehl Ratio computation;
2. The standard star magnitude (m) is extracted from SPH_IRD_TAG_PHOT_TABLE_CALIB according to the filter employed;
3. The exposure time (t) is extracted from the raw header;
4. The zeropoint is calculated as:

$$z = m + 2.5 * \log 10(f/t) \tag{3.1}$$

5. The zeropoint z is corrected by a correction factor c extracted from the appropriate extension of SPH_IRD_TAG_FILTER_TABLE_CALIB:

$$z_c = z - c \tag{3.2}$$

6. Both z and z_c are written in the output products as QC parameters, e.g. for the left field we have ESO QC ZPOINT LEFT and ESO QC ZPOINTCORR LEFT respectively;

If you provide more than one of the optional frames, a choice will be made according to the following priorization:

1. If a SKY_BG_FIT is available, this one is chosen.
2. Else if a SKY_BG is available, this one is chosen.
3. Else if an INS_BG_FIT is available, this one is chosen.
4. Else if an INS_BG is available, this one is chosen.
5. Else if a MASTER_DARK is available, this one is chosen.
6. Else an error will be thrown and execution terminated.

3.2.3.1.6 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.

3.2.4 IRDIS Long-Slit Spectroscopy (LSS) Specific Calibrations and Science

3.2.4.1 sph_ird_wave_calib

3.2.4.1.1 Purpose:

Perform the wavelength calibration

3.2.4.1.2 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	No	1	1
IRD_ MASTER_ DARK	Calibration	Yes	0	1
IRD_ STATIC_ BADPIXELMAP	Calibration	Yes	0	1

3.2.4.1.3 Raw frame keywords used:

none

3.2.4.1.4 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	0,1,2
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	-
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.	1000.0	-
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 [nm].	987.72	-
ird.wave_ calib.wavelength_ line2	double	The wavelength of second line [nm].	1123.71	-
ird.wave_ calib.wavelength_ line3	double	The wavelength of third line [nm].	1309.0	-
ird.wave_ calib.wavelength_ line4	double	The wavelength of fourth line [nm].	1545.07	-
ird.wave_ calib.wavelength_ line5	double	The wavelength of fifth line [nm].	1730.23	-
ird.wave_ calib.wavelength_ line6	double	The wavelength of sixth line [nm].	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

Name	Type	Description	Default	Allowed vals.
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.	3	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.	200	-
ird.wave_ calib.grism_ mode	string	Switch between grism mode (TRUE), prism mode (FALSE), or automatic (AUTO). In auto mode, the ESO.INS1.OPTI1.NAME keyword determines the mode, whether grism or prism. In grism mode the fitting coefficients c2 = c3 = c4 = 0, and the corresponding user parameters are ignored.	AUTO	-
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	-
ird.wave_ calib.c4	double	The c4 coefficient in the fit	82.442	-

3.2.4.1.5 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 recipe will attempt to detect the lines. For this purpose, the image is sliced into lines parallel to the wavelength direction. For each slice, peaks belonging to 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 a linear dispersion and the minimum and maxmimum wavelengths as specified in the header of the master instrument flat field frame) for the maximum image value. The actual positions for all input calibration lines are stored and a polynomial fit of input calibration lines versus acutal pixel positions is performed and used to interpolate all wavelength values between calibration lines for the image slice. Once all slices inside the spectral region have been processed the PDT is updated with the new information and written out as the product.

3.2.4.1.6 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.

3.2.4.2 sph_ird_science_spectroscopy

3.2.4.2.1 Purpose:

Science calibration, spectroscopy mode.

3.2.4.2.2 Input frames:

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

3.2.4.2.3 Raw frame keywords used:

none

3.2.4.2.4 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

3.2.4.2.5 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 before using a user specified method to combine the frames. If a atmospheric calibration is provided this is subtracted from the result.

3.2.4.2.6 Known Issues:

The recipe is not using the wavelength calibration file. In order to removethe effects of the wavelength dependence of the flat, the recipe should reallyuse a series of flats taken at different wavelengths and construct a 'super' flatfrom this using the wavelength calibration file, in the same way as this isdone for IFS. However, it is not clear if this is in fact required or if thewavelength dependence of the flat has too small an effect on the spectroscopicdata reduction to make it necessary to perfom this wavelength dependentcalibration.

3.2.4.2.7 Products:

Name	Type	Description

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

3.2.5 IRDIS Dual-Band Polarimetry (DPI) Specific Calibrations and Science

3.2.5.1 sph_ ird_ science_ dpi

3.2.5.1.1 Purpose:

Science calibration, DPI mode.

3.2.5.1.2 Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IRD_ SCIENCE_ DPI_ RAW	Raw data	No	1	Any
IRD_ MASTER_ DARK	Calibration	Yes	0	1
IRD_ INS_ BG	Calibration	Yes	0	1
IRD_ INS_ BG_ FIT	Calibration	Yes	0	1
IRD_ SKY_ BG	Calibration	Yes	0	1
IRD_ SKY_ BG_ FIT	Calibration	Yes	0	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

3.2.5.1.3 Raw frame keywords used:

none

3.2.5.1.4 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.save_ addprod	bool	Flag signalling whether additional products should be saved. These are the individual, adi combined when required, products for the left and right fields	1	-

Name	Type	Description	Default	Allowed vals.
ird.science_ dpi.save_ interprod	bool	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!	1	-
ird.science_ dpi.outfilename_ left	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	science_ dpi_ left.fits	-
ird.science_ dpi.outfilename_ right	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	science_ dpi_ right.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	-
ird.science_ dpi.use_ sdi	bool	Flag to control usage of SDI	0	-
ird.science_ dpi.keep_ fctable	bool	Has no effect. Use the save_interprod option instead.	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 filter, 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 frame combination. 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

3.2.5.1.5 Description:

This recipe creates the reduced science frames for all science observations with IRDIS in DPI mode. The recipe is essentially identical to the DBI science recipe, except that the final result is saved as as 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.

3.2.5.1.6 Products:

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 polarization (P) image, the badpixels, the rms error and a weightmap of the polarisation. The last 4 extensions contain the same information for the intensity (I) image. Note that when ADI is requested, only 4 extensions are present, containing only the P image.

3.2.6 IRDIS Advanced Recipes for Differential Imaging

These recipes are currently being kept for future use but are not activelymaintained.They should not be used in the present version and will most probably notwork anyway.

3.2.6.1 sph_ird_andromeda

3.2.6.1.1 Purpose:

Andromeda recipe.

3.2.6.1.2 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

3.2.6.1.3 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.

3.2.6.1.4 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 frame combination. A non zero value leads to suppression of high frequencies in the fourier domain before frame combination. 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

3.2.6.1.5 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 obtained from L. Mugnier as much as possible. The basic reduction of raw frames follows that of the other IRDIS science recipes, in particular dark 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.

3.2.6.1.6 Known Issues:

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

3.2.6.1.7 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.

3.2.6.2 sph_ird_loci

3.2.6.2.1 Purpose:

LOCI recipe.

3.2.6.2.2 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

3.2.6.2.3 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.

3.2.6.2.4 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	-
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

Name	Type	Description	Default	Allowed vals.
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 frame combination. A non zero value leads to suppression of high frequencies in the fourier domain before frame combination. 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

3.2.6.2.5 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.

3.2.6.2.6 Known Issues:

No support for radial profile subtraction.

3.2.6.2.7 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.

3.2.7 IRDIS Workflow Summary

The IRDIS workflow is summarized in Fig.3.3(initial part) and Fig.3.4(final part).

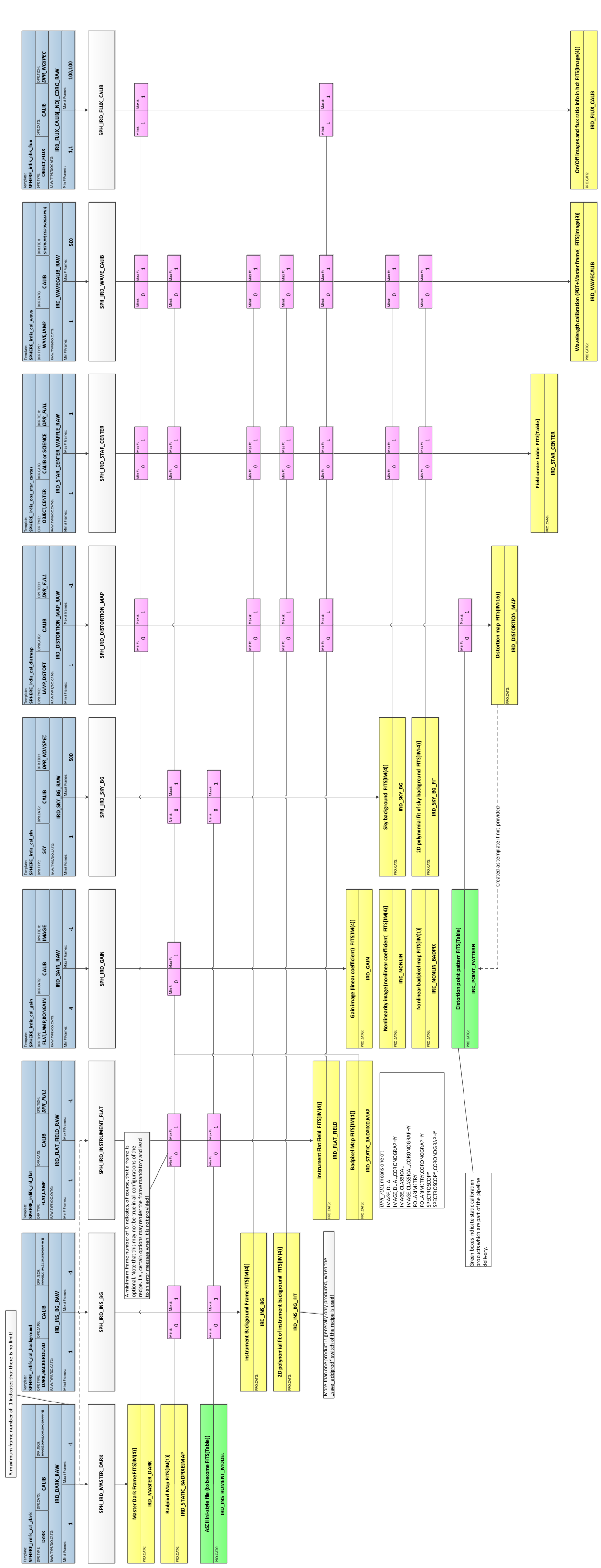

Figure 3.3: Initial part of the IRDIS calibration cascade (workflow).

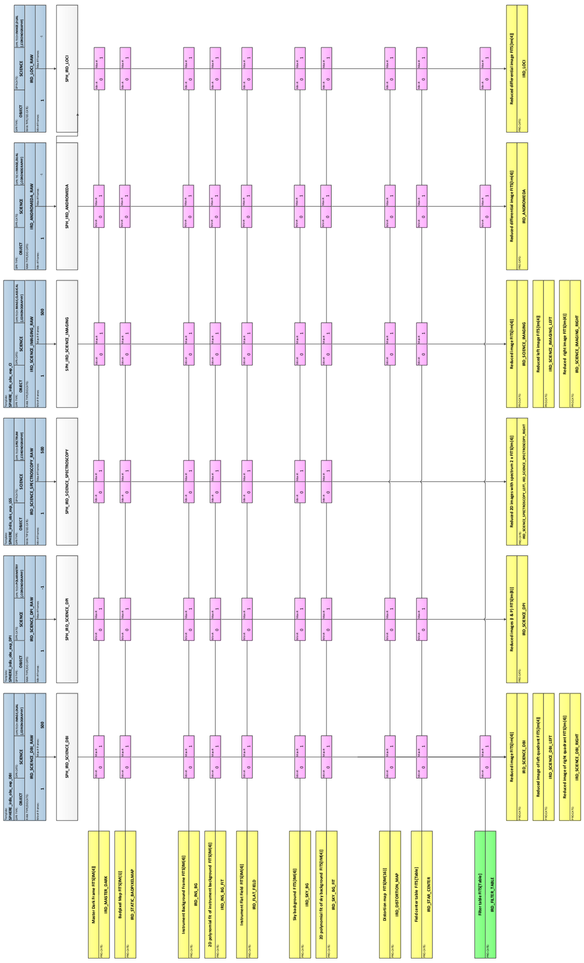

Figure 3.4: Final part of the IRDIS calibration cascade (workflow)

3.2.8 IRDIS Special notes

3.2.8.1 The Instrument Model

Many IRDIS recipes list IRD_INSTRUMENT_MODEL frames as possible input.These are currently unavailable and and the instrument model, a set ofnumbers that describe the positions and extents of the two FOVs on theIRDIS detector are usually stored in IRD_FLAT_FIELD frames produced bythe sph_ird_instrument_flat recipe.This is the reason why IRD_FLAT_FIELD frames are usually mandatory in allhigher-order recipes.

3.2.8.2 Dark and Background Calibration

Most IRDIS recipes accept several types of dark or background products forsubtraction:

- IRD_MASTER_DARK
- IRD_INS_BG
- IRD_INS_BG_FIT
- IRD_SKY_BG
- IRD_SKY_BG_FIT

Some fo the recipes require that at least one dark is provided, some runalso without any dark or background frame.For details, look at the recipe descriptions in the subsections above orat the recipe man pages.If more than one dark / background frame are fed into a single recipe,the recipe will select one according to the following logic:

1. if IRD_SKY_BG_FIT is provided, this one is chosen
2. else if IRD_SKY_BG is provided, this one is chosen
3. else if IRD_INS_BG_FIT is provided, this one is chosen
4. else if IRD_INS_BG is provided, this one is chosen
5. else if IRD_MASTER_DARK is provided, this one is chosen
6. else an error is thrown, some recipes also continue without subtractingany dark or background.

Note that the else-if chain implies that all otheres are ignored after achoice has been made.For all dark/background products a pre-selection should be made by theuser (or any automatic environment) to select only calibration productswith the same DIT and readout mode.For everything axcept the IRD_MASTER_DARK frames, also the filter configurationshould be matched!

3.3 IFS

IFS offers two basic modes:

- YH
- YK

The calibration cascade (workflow) is identical for both modes.Care should however be taken that calibration products and raw frames alwayswere taken in the same mode!

3.3.1 IFS Calibrations and Science

3.3.1.1 sph_ifs_gain

3.3.1.1.1 Purpose:

Measure the detector gain

3.3.1.1.2 Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ GAIN_ RAW	Raw data	No	4	Any

3.3.1.1.3 Raw frame keywords used:

none

3.3.1.1.4 Parameters:

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	-

3.3.1.1.5 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 should be different either by increasing the intensity of the illumination source or by using different exposure times. 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, which corrects 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 algorithms to calculate the gain, one straightforward fitting algorithm and a more complex fitting algorithm that takes the correct number of fowler samples into account. The second algorithm is switched on using the vacca user parameter and is preferable 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 when an 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 all input 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 variance image has been determined the median of the mean image and the corresponding variance is taken as one data point. The collection of input cubes then lead to a collection of data points of median and variance, giving measurements of the variance vs. median relation for the detector. This is then fitted using a polynomial of the specified order (usually 1 or 2). The slope of this curve is the inverse of the gain while the offset gives an estimate of the read out noise. Note that the read out noise estimate 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 ron are 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 noise properties 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 product FITS file. Note that the pixel-by-pixel gain values are often very noisy and can not be used to obtain precise 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 (quadratic) coefficient of the individual pixel fits is saved in an additional FITS file. All pixels that have second order quadratic 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.

3.3.1.1.6 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.

Name	Type	Description
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.

3.3.1.2 sph_ifs_master_dark

3.3.1.2.1 Purpose:

Creation of the master dark frame

3.3.1.2.2 Input frames:

Data Type (TAG)	Source	Optional	Min	Max
IFS_ DARK_ RAW	Raw data	No	1	Any

3.3.1.2.3 Raw frame keywords used:

none

3.3.1.2.4 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
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

Name	Type	Description	Default	Allowed vals.
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	-

3.3.1.2.5 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 collapse algorithm specified (usually the clean_mean algorithm). After all input frames are combined in this way, the badpixels are determined on 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 subtracted from 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 the specified 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) pixels for each output pixel. The hotpixel map is also written out as a separate parameter.

3.3.1.2.6 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.
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.

3.3.1.3 sph_ifs_master_detector_flat

3.3.1.3.1 Purpose:

Creation of the master detector flat frame

3.3.1.3.2 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

3.3.1.3.3 Raw frame keywords used:

none

3.3.1.3.4 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	-
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

Name	Type	Description	Default	Allowed vals.
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

3.3.1.3.5 Description:

The detector flat field recipe for IFS is very similar to the instrument flat field 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 and flat 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 large scale flat field which is a smoothed flat field and hence only shows large scale structures, and a normal flat field. Experience from the SPHERE Data Centre shows that the IFS_MASTER_DFF_LONG files (both at the various wavelengths and the white one) are sufficient to correct the pixel-to-pixel variation in the detector response. The preamplifier correction flat and the large scale flat field (which contains smoothed bad pixels) are not needed for further processing. 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 derived from the input raw data (note that it is 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 for all 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 data m_i, v_i. The resulting slope represents the response of an individual 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 method or 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) can be expected, but does not allow anything but linear fits and can hence not be used to assess detector non-linearity. 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 user parameters 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 above a 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 created by smoothing the flat field with either a gaussian kernel using FFT or a square kernel using the specific CPL filter algorithm. Unless you know what you are doing leave the default method here which is the FFT smoothing.

3.3.1.3.6 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_ PREAMP_ FLAT	FITS[Im(4)]	Optional product with a preamp correction flat, formatted as above.
IFS_ LARGE_ SCALE_ FLAT	FITS[Im(4)]	Optional product with a large scale structure flat, formatted as above.
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.

3.3.1.4 sph_ifs_spectra_positions

3.3.1.4.1 Purpose:

Determinate of the spectra regions on detector

3.3.1.4.2 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_ CAL_ BACKGROUND	Calibration	Yes	0	1
IFS_ MASTER_ DARK	Calibration	Yes	0	1
IFS_ LENSLET_ MODEL	Calibration	Yes	0	1

3.3.1.4.3 Raw frame keywords used:

none

3.3.1.4.4 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-JH (TRUE) instead of Y-J (FALSE). Note that this parameter is only effective if no input IFS lenslet model frame is given and the header parameter is set to false.	0	-

Name	Type	Description	Default	Allowed vals.
ifs.spectra_ positions.header	bool	Flag to set whether to try and set parameters automatically from keywords given in the header of the input files.	1	-
ifs.spectra_ positions.correct_ nonlin	bool	Apply non-linear correction to wavelength calibration as discovered necessary by D. Mesa in June 2015	1	-

3.3.1.4.5 Description:

This recipe associates the IFS spectra with lenslets and associates pixels with wavelengths. The wavelengths are an initial estimation from the lenslet model and the sph_ifs_wave_calib should be used to refine them. The raw frames are reduced by optionally dark subtracting either a master background or a master dark and by flat fielding and then combined using the combination algorithm chosen (usually the clean mean algorithm). The flat field used can be either a detector flat field, a flat field of the whole instrument (detector+IFU) or any other flat deemed to be useful for this purpose. In most cases a detector flat field obtained with the broad band calibration lamp seems the best choice. After a combined and reduced frame has been produced the following algorithm is applied:

- A thresholding algorithm will determine the spectra regions from the combined frame. From the regions a point pattern is calculated. A point pattern is a list of points. In this case the points are the centers of the regions;
- Calculate the average width of the spectra regions and use it to scale the width and the stretch factors (x and y directions) of the IFS lenslet model;
- A second point pattern is then determined using the (scaled) IFS lenslet model. If a IFS lenslet model was provided in the form of a FITS file with the model parameters as keywords in its header, this model is used, otherwise the default model is used, please see the description of the lenslet model in the section on static calibrations;
- The expected point pattern is compared with the actual one to determine a relative scale and an offset;
- The IFS lenslet model used for the expected pattern is subsequently scaled and shifted accordingly to reflect the new, actual values of scale and position.
- If desired, the distortion is calculated. This may be crucial to ensure correct wavelength calibrations later in the calibration cascade. If distortion is allowed (using the ifs.spectra_positions.distortion parameter) then the recipe will also fit a 2D polynomial of 4th order in x and y directions to the difference between measured and predicted point patterns. The coefficients of this polynomial model are incorporated in the IFS_SPECPOS product and will be used subsequently in all recipes making use of the created lenslet model file.
- For each pixel having coordinates (x, y) an initial wavelength estimation is calculated. For each point the polygon it belongs to is calculated according to the refined model. The polygon can be rotated, therefore a new aligned polygon having the same center, width and height is calculated. The lower y coordinate y_{low} of the aligned polygon is used to calculate the corresponding wavelength $\lambda(y)$:

$$\lambda(y) = (y - 0.5 - y_{low}) \cdot d + \lambda_{min} \tag{3.3}$$

where d is the model dispersion and λ_{min} is the minimum wavelength according to the model;

- A non linear correction is then applied to $\lambda(y)$ if ifs.spectra_positions.correct_nonlin = TRUE:
$$\lambda'(y) = a \cdot \lambda^2(y) + (b+1) \cdot \lambda(y) + c \quad (3.4)$$
where a, b and c are 0.30870, -0.74312, 0.41505 in H mode and 0.67754, -1.4464, 0.75754 in Y mode respectively;

Finally the calculated data (mode, wavelengths, distortion) is used to construct a pixel description table which is saved, together with the corrected lenslet model parameters, as the primary product of this recipe.

3.3.1.4.6 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.

3.3.1.5 sph_ifs_instrument_flat

3.3.1.5.1 Purpose:
Determine the full instrument flat field OR the IFU flat

3.3.1.5.2 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_ CAL_ BACKGROUND	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

3.3.1.5.3 Raw frame keywords used:
none

3.3.1.5.4 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.	1	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	-

3.3.1.5.5 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 and the input files available. In the total flat field mode the recipe reads in the spectra positions file to set the illuminated regions. The recipe then constructs a flat field in the same way as done for sph_ifs_master_detector_flat (optionally first subtracting either a master background or a master dark). The resulting frames is then saved as the total instrument flat. Note that this frame has 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 positions table. First the same steps are carried out as for the total flat mode. However, the wavlength calibration file is then used to first construct a wavelength dependent flat field (also sometimes called a super flat field), making use of 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 the header of the input wavelength calibration frame. This lenslet description table now contains the extracted spectra data for all lenslets. These are then collapsed along the wavelength direction (taking the median values) to obtain a flat field value for all lenslets. 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 flat field values for all lenslets. This table is used in other recipes when the IFU flat field is to be applied.

3.3.1.5.6 Products:

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.

Name	Type	Description
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.

3.3.1.6 sph_ifs_wave_calib

3.3.1.6.1 Purpose:

Create the wavelength calibration data

3.3.1.6.2 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_ CAL_ BACKGROUND	Calibration	Yes	0	1
IFS_ MASTER_ DARK	Calibration	Yes	0	1

3.3.1.6.3 Raw frame keywords used:

none

3.3.1.6.4 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
ifs.wave_ calib.number_ lines	int	The number of wavelength lines to fit. A number of zero (default) means automatic setting. In that case the number of wavelengths and the wavelengths themselves are set automatically from the header of the pixel description table.	0	0-5
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

Name	Type	Description	Default	Allowed vals.
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.	1	-
ifs.wave_ calib.fit_ window_ size	int	Half 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
ifs.wave_ calib.save_ addprod	bool	Flag to signal whether additional products - in this case the wavelength calibrated cube - should be saved.	0	-

3.3.1.6.5 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 no 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). Optionally either a master background or a master dark dark is subtracted and the result is 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). The i-th spectrum has wavelengths $w^{(i)}(y)$ and flux $s^{(i)}(y)$ where y is the pixel index in the wavelength direction. Around each of the line l with wavelength w_l as specified in the user input, the flux weighted mean position is determined in a window around the expected position with a width as specified by the fit_window_size parameter:

$$y'_l = \frac{\sum_{y=y_{min}}^{y_{max}} s(y)^2 \cdot y}{\sum_{y=y_{min}}^{y_{max}} s(y)^2} \tag{3.5}$$

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 chosen 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 predicted pixel for the wavelength w_l. Once parameters y'_l

have been determined for all $l = 1..n_l$ for a spectrum, a polynomial 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. More specifically, for the i-th spectrum a vector of wavelengths $w_c^{(i)}(y)$ is generated associating the y-th pixel with the corresponding wavelength. 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 association was found). In case no spectrum could be extracted in the first place, all wavelengths are set to zero and all pixels in the spectra region thereby marked as bad. Hence, ignoring the spectra marked as bad regions, for the i-th spectrum the final wavelengths vector $w_f^{(i)}$ is:

$$w_f^{(i)} = \begin{cases} w^{(i)} & \text{if } w_c^{(i)} \text{ deviates too much from } w^{(i)} \\ w_c^{(i)} & \text{otherwise} \end{cases} \tag{3.6}$$

And for every i-th spectrum the following QC values are calculated:

- $w_{min}(i) = \min w_f^{(i)}$
- $w_{MAX}(i) = \max w_f^{(i)}$
- $w_{cent}(i) = w_f^{(i)}(\frac{N-1}{2})$

Finally the resolving power $r_p^{(i)}(y)$ of the i-th spectrum is calculated as follows:

$$r_p^{(i)}(y) = \frac{w_f^{(i)}(y) + w_f^{(i)}(y+1)}{2(w_f^{(i)}(y+1) - w_f^{(i)}(y))} \tag{3.7}$$

lastly the median resolving power for the i-th spectrum is:

$$r(i) = \text{median}(r_p^{(i)}) \tag{3.8}$$

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. In particular:

- "ESO QC MEDIAN DISPERSION" = d_m = median(d), where d is the dispersion image related to the output pixel description table;
- "ESO QC MEDIAN RESOLVING POWER" = median($r(i)$).
- "ESO DRS MEDIAN MIN WAVEL" = median($w_{min}(i)$) $- \frac{d_m}{2}$
- "ESO DRS MEDIAN MAX WAVEL" = median($w_{MAX}(i)$) $+ \frac{d_m}{2}$;
- "ESO DRS MEDIAN CENT WAVEL" = median($w_{cent}(i)$).

As of SPHERE 0.42.0b, an additional resolving power calculation is performed as follows:

- A median collapse spectrum is formed from the `IFS_WAVECALIB_CUBE` product by computing the median of the good pixels in each plane of the cube (excluding the 60 pixels closest to each image edge);
- A Gaussian fit is made to each of the laser lamp arc lines in the median collapse spectrum (3 lines in YJ, 4 in YJH). The fit is made in flux as a function of pixel coordinate along the spectrum;

- A linear fit is then made between the known wavelengths of the laser lamp arc lines as a function of the fitted pixel coordinate of those lines.

From these calculations, the following header keywords are populated and saved in both the `IFS_WAVECALIB` and `IFS_WAVECALIB_CUBE` products:

- The integrated flux of laser line n is stored as `ESO QC FLUX LASERn`;
- The resolving power of laser line n (i.e., λ_0/Δ_λ, where λ_0 is the nominal wavelength of the laser lamp and Δ_λ is the FWHM of the Gaussian fit) is stored as `ESO QC RESPOW LASERn`;
- The wavelength-to-pixel scale, i.e. the gradient of the linear fit described above, is stored as `ESO QC DISPERSION`;
- The RMS value of the linear fit described above (formally, the coefficient of determination, commonly referred to as R^2), is stored as `ESO QC DISPERSION RMS`.
- The wavelength in pixel 1 (i.e. the first pixel), as determined by the linear fit described above, is stored as `ESO QC MIN LAMBDA`.

3.3.1.6.6 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.
IFS_ WAVECALIB_ CUBE	FITS[Imcube(4)]	The wavelength calibrated cube.

3.3.1.7 sph_ifs_star_center

3.3.1.7.1 Purpose:

Determine the field centre

3.3.1.7.2 Input frames:

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

3.3.1.7.3 Raw frame keywords used:

none

3.3.1.7.4 Parameters:

Name	Type	Description	Default	Allowed vals.
ifs.star_center.outfilename	string	The output filename for the product. Please also see the esorex documentation for naming of output products.	star_center.fits	-
ifs.star_center.coll_alg	int	The collapse algorithm to use. 0 = Mean, 1 = Median.	1	0,1
ifs.star_center.clean_mean.reject_high	int	The clean mean reject pixels on high end.	1	0-20
ifs.star_center.clean_mean.reject_low	int	The clean mean reject pixels on low end.	1	0-20
ifs.star_center.sigma	double	The sigma threshold to use for source detection	10.0	-
ifs.star_center.use_waffle	bool	Flag to whether to expect a waffle image (4 images in cross formation) or not (single central fit).	1	-
ifs.star_center.qc	bool	If set QC output for this recipe is produced.	0	-
ifs.star_center.save_interprod	bool	Flag to signal if intermediate products should be kept	0	-
ifs.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	-
ifs.star_center.qc_window_x	int	This specifies the window width for QC output in pixels.	256	-
ifs.star_center.qc_window_y	int	This specifies the window height for QC output in pixels.	256	-

3.3.1.7.5 Description:

This recipe creates a table with centre star positions. The input raw frames are 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 dimensions 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 extracted and left and right part are seperately analysed using a aperture detection algorithm. The aperture detection algorithm detects all connected regions of 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 coronograph is out and no waffle stars are formed). After frame centers have been determined for all waffle images an internal table is created with an entry for each waffle image, giving the time of the start of the exposure and the centre information. The recipe reads the position of the IFS 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.

3.3.1.7.6 Known Issues:

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

3.3.1.7.7 Products:

Name	Type	Description
IFS_ 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.

3.3.1.8 sph_ifs_science_dr

3.3.1.8.1 Purpose:

Reduce science observations

3.3.1.8.2 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_ CAL_ BACKGROUND	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

3.3.1.8.3 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.

3.3.1.8.4 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 when creating a product with ADI. 0 = Mean, 1 = Median, 3 = Weighted mean.	3	0,1,3
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	-
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. Note that the parameters to the ADI algorithm are fixed - it uses an FFT transform with no filter.	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	-

Name	Type	Description	Default	Allowed vals.
ifs.science_ dr.spec_ deconv_ filename	string	The basename for the spectra deconvolution output files (without 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. [Currently fixed at 5 when ifs.science_dr.spec_deconv = TRUE]	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. [Currently fixed at FALSE when ifs.science_dr.spec_deconv = TRUE]	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. Unused if ifs.science_dr.spec_deconv = TRUE as user_cent is then fixed to FALSE.	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. Unused if ifs.science_dr.spec_deconv = TRUE as user_cent is then fixed to FALSE.	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. [Currently fixed at 1.0 when ifs.science_dr.spec_deconv = TRUE]	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. [Currently fixed at 2.0 when ifs.science_dr.spec_deconv = TRUE]	-1.0	-
ifs.science_ dr.badpixco.apply	bool	Flag to set the application of the automatic badpixel correction	0	-
ifs.science_ dr.xtalkco.apply	bool	Flag to set the application of the cross talk correction	0	-
ifs.science_ dr.xtalkco.largescale.apply	bool	Flag to set the application of the large scale crosstalk correction. If set to false, only the small-scalecrosstalk gets corrected which usually yields better results.	0	-
ifs.science_ dr.xtalkco.sepmax	int	The sliding correction window half-size (the window size is 2 * sepmax + 1)	20	-

Name	Type	Description	Default	Allowed vals.
ifs.science_ dr.xtalkco.bfac	double	The parameter bfactor in the small talk cross talk correction. To correct values, the central subwindow pixel value times 1.0/(1.0 + pow(rdist/bfac,powfac)) is subtracted, with rdist the pixel distance to the central subwindow pixel.	0.40443667	-
ifs.science_ dr.xtalkco.powfac	double	The parameter powfac in the small talk cross talk correction. To correct values, the central subwindow pixel value times 1.0/(1.0 + pow(rdist/bfac,powfac)) is subtracted, with rdist the pixel distance to the central subwindow pixel.	3.0	-
ifs.science_ dr.xtalkco.lgscalewin	int	The large scale cross talk correction subwindow size.	64	-
ifs.science_ dr.xtalkco.threshold	double	The large scale crosstalk correction threshold to use. Any values that are in between -threshold and +threshold will be set to the median image value.	50000.0	-
ifs.science_ dr.xtalkco.smoothing	double	The large scale crosstalk correction threshold smoothing fwhm to use, This is used to to smooth the image before subtracting from the original.	100.0	-
ifs.science_ dr.xtalkco.stephist	double	The bin size to use for the creation of the pixel histogram for the most likely value find. The most likely value is found in each subwindow and the smmothed version of these subtracted from the originial.	20.0	-
ifs.science_ dr.badpixco.threshold	double	The absolute threshold for badpixel correction to use. This is applied in a logical AND operation with the badpixel correction factor threshold. Any pixels above this value, that are also above badpixco.fac * median, are set to the median of the surrounding 8 pixels.	100.0	-
ifs.science_ dr.badpixco.fac	double	The absolute threshold factor for badpixel correction to use. This is applied in a logical AND operation with the absolute badpixel correction threshold. Any pixels above badpixco.fac * median AND also above the absolute threshold, are set to the median of the surrounding 8 pixels.	5.0	-
ifs.science_ dr.badpixco.border	int	The border size in pixel to ignore for the purpose of badpixel correction.	100	-

3.3.1.8.5 Description:

This is the science calibration recipe for IFS. The input raw observation frames are reduced by dark subtracting each one, if a master background or alternatively a master dark frame is given as input. Also, in case a pre-amplifier correction frame (basically for de-striping) the raw frames are 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 to wavelength correspondence into account using the information from the input wavelength calibration file. After dividing out the super flat, a broad band flat field that should have been created from recent 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 field recipe 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. To combine frames manually, the recipe the use_adi and spec_deconv flags should be set to false. Note that the ADI and spectral deconvolution routines currently use fixed parameters as noted. The correction of (detector pixel-to-pixel) correction can be switched on by the appropriate flag. It is strongly recommended to use only the small-scale correction. The large-scale variant is a bit over-aggressive and deprecated. The correction of detector-level badpixels can also be switched on with a flag. This will interpolate over badpixels to substitute their values instead of relying on dithering.

3.3.1.8.6 Products:

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

3.3.1.9 sph_ifs_detector_persistence

3.3.1.9.1 Purpose:

Measure the detector persistence.

3.3.1.9.2 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

3.3.1.9.3 Raw frame keywords used:

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

3.3.1.9.4 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	-

Name	Type	Description	Default	Allowed vals.
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	-

3.3.1.9.5 Description:

This recipe determines the detector persistence, by measuring the signal fall-off rate. 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 containing illuminated (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, a simple thresholding algorithm is used on the illuminated but **unsaturated** image to determine illuminated and unilluminated pixel sets, P_i and P_u. For each of the unilluminated frames,the mean for the unilluminated pixels $< P_u >$ is subtracted 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 $P(t) = c_0 + c_1 \times 1/t + c_2 \times 1/t^2 + \ldots$. Up to which coefficient the fit 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.

3.3.1.9.6 Products:

Name	Type	Description

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).

3.3.1.10 sph_ifs_cal_background

3.3.1.10.1 Purpose:

Measure the instrument background.

3.3.1.10.2 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

3.3.1.10.3 Raw frame keywords used:

none

3.3.1.10.4 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	The collapse algorithm to use. 0 = Mean, 1 = Median, 2 = 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.	0	-
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

Name	Type	Description	Default	Allowed vals.
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.	0.0	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

3.3.1.10.5 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 be subtracted from the resulting frame if porovided. Static badpixels are determined using the sigma_clip user 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 dark was 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 badpixels have been interpolated). If so desired, a 2D polynomial fit to the measured (possibly smoothed) bacground is also written out as a second product.

3.3.1.10.6 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.

3.3.1.11 sph_ifs_distortion_map

3.3.1.11.1 Purpose:

Measure the lenslet array distortion

3.3.1.11.2 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_CAL_BACKGROUND	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

3.3.1.11.3 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

3.3.1.11.4 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, 2 = Clean mean	1	0,1,2
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

Name	Type	Description	Default	Allowed vals.
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	-
ifs.distortion_ map.add_ rotation	double	Additional rotation angle that the measured point pattern is rotated by before attempting a match. This is applied in addition to the rotations specified in the lenslet model in the keywords of the calibration input files. The sense of direction is specified in the same sense as in the lenslet model.	0.0	-360.0-360.0

3.3.1.11.5 Description:

This recipe measures the distortion of the lenslet grid. The input raw frames are first reduced in the same way as for the science_dr recipe (allowing dither but no frame rotation), optionally including dark subtraction via either a master background or a master dark and optionally (if all flats provided) flat fielding. The resulting monochromatic images are then collapsed along the wavelength direction giving a single image. This image is then analysed to detect the point sources using simple thresholding, 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 from the reduced images detected point sources. This will then be written out as a product. If a input point pattern was provided, the recipe will skip this step and proceed directly to the measurement of the distortion. Please note that the distortion is measured in any case which means that the output distortion should be zero in case that no input point pattern was provided. The distortion map is constructed by comparing the detected point sources position with the expected point source position as provided in the input point pattern. Each comparison is done with the closest point found and yields one distortion vector. Upon completion, all distortion vectors that have a length larger than the max_distortion specified as a user parameter will be removed before the x and y components of the distortion vectors 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 a self 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 inspected to verify that the distortion map has been calculated with sufficient accuracy in the detector region(s) of interest.

3.3.1.11.6 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.

Name	Type	Description
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.

3.3.2 IFS Workflow Summary

The IFS workflow is summarized in Fig.3.5

3.3.3 IFS Special Notes

The IFS calibration cascade is currently under heavy revision.It is thus actually not recommended to draw information on the actual workflowfrom this version of the manual! the individual recipe descriptions aboveare up to date and accurate, but the overall workflow will change considerableover the next weeks!

3.4 ZIMPOL

ZIMPOL offers the following basic observing modes:

- Imaging
- P1 polarimetry
- P23 polarimetry

These have their own science recipes and a number of dedicated calibrations.Additionally, ZIMPOL has different operational modes:

- FastPol
- SlowPol
- Window

These do not affect the choice or the order of the calibration workflow,but data taken in any of these sub-modes should not be mixed with datafrom other sub-modes in a single cascade!

3.4.1 ZIMPOL Imaging Calibrations and Science

These are calibration that apply to the IMAGING mode of ZIMPOL.

3.4.1.1 sph_zpl_preproc_imaging

3.4.1.1.1 Purpose:

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

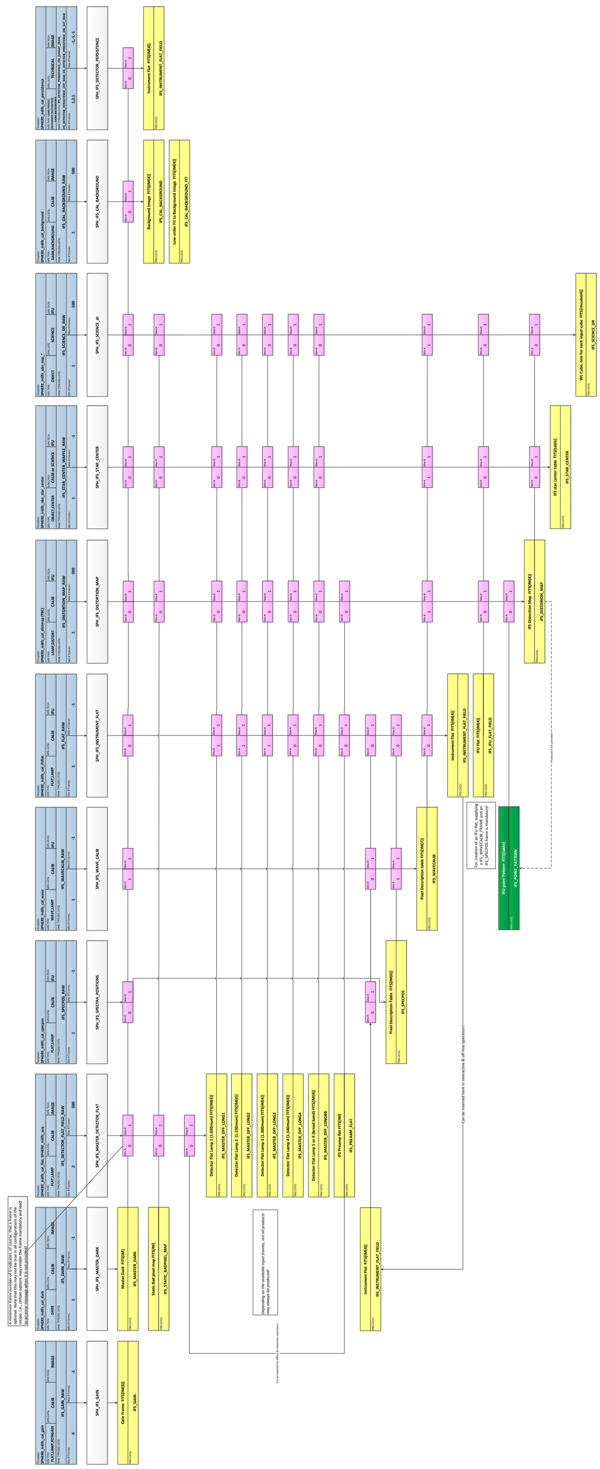

Figure 3.5: Summary of the IFS calibration cascade (workflow)

3.4.1.1.2 Input frames:

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

3.4.1.1.3 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

3.4.1.1.4 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	-

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

3.4.1.1.5 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 computed mean 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) with the 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.

3.4.1.1.6 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

3.4.1.2 sph_zpl_master_bias_imaging

3.4.1.2.1 Purpose:

Create master bias, imaging mode.

3.4.1.2.2 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

3.4.1.2.3 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.

3.4.1.2.4 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	-

Name	Type	Description	Default	Allowed vals.
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	-
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	-

3.4.1.2.5 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-processed 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 will not 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 and its rms value set to the 1e10 in the rms-map). Therefore, using the master bias for all subsequent imaging recipes in the default 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.

3.4.1.2.6 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;

Name	Type	Description
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;
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;

3.4.1.3 sph_zpl_master_dark_imaging

3.4.1.3.1 Purpose:

Create master dark, imaging mode.

3.4.1.3.2 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

3.4.1.3.3 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.

3.4.1.3.4 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	-
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	-

3.4.1.3.5 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 subtracted by 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.

3.4.1.3.6 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.

Name	Type	Description
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.
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.

3.4.1.4 sph_zpl_intensity_flat_imaging

3.4.1.4.1 Purpose:

Create intensity flat field, imaging mode.

3.4.1.4.2 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

3.4.1.4.3 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<<.

3.4.1.4.4 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	-
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	-

Name	Type	Description	Default	Allowed vals.
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	-
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	-

3.4.1.4.5 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-frames individually) 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 then applied 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 for all 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 data m_i,v_i.
4. The fit itself is performed either using a maximum likelyhood method or 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) can be 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 user parameters 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 above a 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 badpixels that are stored in the master flat field itself as produced by this recipe contain all the badpixels (for each sub-frames individually) at this point in the cascade. Pixels that were 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 the flat field recipe itself deemed to be bad. This does not necessarily include all the badpixels from 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.

3.4.1.4.6 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.
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.

Name	Type	Description
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).

3.4.1.5 sph_zpl_star_center_img

3.4.1.5.1 Purpose:

Determine the center of the star center frame, imaging mode.

3.4.1.5.2 Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ STAR_ CENTER_ IMG_ RAW	Raw data	Yes	0	Any
ZPL_ STAR_ CENTER_ IMG_ PREPROC	Calibration	Yes	0	Any
ZPL_ STAR_ CENTER_ IMG_ PREPROC_ CAM1	Calibration	Yes	0	Any
ZPL_ STAR_ CENTER_ IMG_ 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

3.4.1.5.3 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]

3.4.1.5.4 Parameters:

Name	Type	Description	Default	Allowed vals.
zpl.star_ center_ img.outfilename	string	The output filename for the product, camera-1/2. Please also see the esorex documentation for naming of output products.	zpl_ star_ center_ img.fits	-
zpl.star_ center_ img.outfilename_ cam1	string	The output filename for the product, camera-1. Please also see the esorex documentation for naming of output products.	zpl_ star_ center_ img_ cam1.fits	-
zpl.star_ center_ img.outfilename_ cam2	string	The output filename for the product, camera2. Please also see the esorex documentation for naming of output products.	zpl_ star_ center_ img_ 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.star_ center_ img.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.star_ center_ img.filter_ radius	double	Filter radius for the frame combination. A non zero value leads to suppression of high frequencies in the fourier domain before frame combination. 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.star_ center_ img.sigma	double	The sigma threshold to use for source detections	10.0	-
zpl.star_ center_ img.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	-
zpl.star_ center_ img.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.star_ center_ img.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.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	-

3.4.1.5.5 Description:

The recipe determines the position of star centers for the imaging modes. The input frames might be either science imaging raw frames with the ZPL_STAR_CENTER_IMG_RAW tag or pre-processed science imaging frames, which should carry the ZPL_STAR_CENTER_IMG_PREPROC_CAM1 and/or ZPL_STAR_CENTER_IMG_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, 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_imaging for the detailed description of the pre-processing step). The pre-processed star center 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 then saved as intermediate products. The next step is to combine all these calibrated and de-dithered 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 star-center image, badpixel-map, ncomb-map and rms-map. - combined star-center dark image (dark current), badpixel-map, ncomb-map and rms-map. The combined star center frame is then analysed using an aperture detection algorithm: - The aperture detection algorithm detects all connected regions of 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 coronograph is out and no waffle stars are formed). The coordinates of the detected frame centers are added as the keywords in the header of the output fits-file products for each camera: - DRS ZPL STAR CENTER IFRAME XCOORD; - DRS ZPL STAR CENTER IFRAME YCOORD; - DRS ZPL STAR CENTER PFRAME XCOORD; - DRS ZPL STAR CENTER PFRAME YCOORD. This star center calibrated products for each camera should be used in the science imaging recipe.

3.4.1.5.6 Products:

Name	Type	Description
ZPL_ STAR_ CENTER_ IMG	FITS[Im(8)]	The final combined star center frame of the DOUBLE IMAGING format. This frame contains 8 image extensions: - combined intensity image, badpixel-map, ncomb-map and rms-map; - combined dark image (dark current), badpixel-map, ncomb-map and rms-map.
ZPL_ STAR_ CENTER_ IMG_ CAM1	FITS[Im(8)]	The final combined star center frame of the DOUBLE IMAGING format for the camera-1. This frame contains 8 image extensions: - combined intensity image, badpixel-map, ncomb-map and rms-map; - combined dark image (dark current), badpixel-map, ncomb-map and rms-map.
ZPL_ STAR_ CENTER_ IMG_ CAM2	FITS[Im(8)]	The final combined star center frame of the DOUBLE IMAGING format for the camera-2. This frame contains 8 image extensions: - combined intensity image, badpixel-map, ncomb-map and rms-map; - combined dark image (dark current), badpixel-map, ncomb-map and rms-map.

3.4.1.6 sph_zpl_science_imaging

3.4.1.6.1 Purpose:

Reduce science frames in the imaging modes.

3.4.1.6.2 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_ STAR_ CENTER_ IMG_ CAM1	Calibration	Yes	0	1
ZPL_ STAR_ CENTER_ IMG_ CAM2	Calibration	Yes	0	1
ZPL_ STAR_ CENTER_ IMG	Calibration	Yes	0	1
ZPL_ FIELD_ CENTER_ TABLE	Calibration	Yes	0	Any
ZPL_ FILTER_ TABLE	Calibration	Yes	0	1
ZPL_ PHOT_ STAR_ TABLE	Calibration	Yes	0	1

3.4.1.6.3 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)

3.4.1.6.4 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_ 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.science_ imaging.coll_ alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median. Default is 0 = Mean.	0	0,1
zpl.science_ imaging.filter_ radius	double	Filter radius for the frame combination. A non zero value leads to suppression of high frequencies in the fourier domain before frame combination. 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 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_ imaging.star_ center_ iframe	bool	Flag to set if only the center coordinates of the iframe from the star center calibration frame should be used as a center coordinates to de-rotate iframe and pframe[dark current] (TRUE)	1	-
zpl.science_ imaging.center_ xoffset_ cam1	double	X-offset from the center of the image for the camera-1	0.0	-512.0-512.0
zpl.science_ imaging.center_ yoffset_ cam1	double	Y-offset from the center of the image for the camera-1	0.0	-512.0-512.0

Name	Type	Description	Default	Allowed vals.
zpl.science_ imaging.center_ xoffset_ cam2	double	X-offset from the center of the image for the camera-2	0.0	-512.0-512.0
zpl.science_ imaging.center_ yoffset_ cam2	double	Y-offset from the center of the image for the camera-2	0.0	-512.0-512.0
zpl.science_ imaging.star_ r	double	The star radius [arcsecond] used for the Strehl ratio estimate. A negative value disables the estimation. When AO is enabled and 0 (default) is provided 2 arcseconds are used. When AO is disabled and 0 is provided a radius corresponding to 500 PIXEL is used. This option is ignored in absence of a ZPL_FILTER_TABLE frame.	0.0	-
zpl.science_ imaging.bg_ r1	double	The internal radius [arcsecond] of the background used for the Strehl ratio estimate. When AO is enabled and 0 (default) is provided 2 arcseconds are used. When AO is disabled and 0 is provided a radius corresponding to 500 PIXEL is used. This option is ignored in absence of a ZPL_FILTER_TABLE frame.	0.0	-
zpl.science_ imaging.bg_ r2	double	The external radius [arcsecond] of the background used for the Strehl ratio estimate. When AO is enabled and 0 (default) is provided 3 arcseconds are used. When AO is disabled and 0 is provided a radius corresponding to all the PIXELS in the image is used. This option is ignored in absence of a ZPL_FILTER_TABLE frame.	0.0	-
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	-

3.4.1.6.5 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/or ZPL_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. Note: the calibration frames with SPH_ZPL_TAG_STAR_CENTER_IMG_CALIB_CAM1(CAM2) tags provide the center coordinates to rotate around. If these calibrations are not presented the center of the frames will be used (normally, xc=yc=512 pixel). 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. Some additional notes:

- For a point-source an estimate of the Strehl ratio may be useful. The presence of a filter table frame will enable the estimation, which on failure will do nothing and on success will insert Strehl related QC parameters for both cameras into the product headers related to the second camera.

3.4.1.6.6 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.

3.4.2 ZIMPOL Polarimetry Calibrations and Science

3.4.2.1 sph_zpl_preproc

3.4.2.1.1 Purpose:

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

3.4.2.1.2 Input frames:

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

3.4.2.1.3 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

3.4.2.1.4 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	-

3.4.2.1.5 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 computed mean 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.

3.4.2.1.6 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

3.4.2.2 sph_zpl_master_bias

3.4.2.2.1 Purpose:

Create master bias, polarization modes.

3.4.2.2.2 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

3.4.2.2.3 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.

3.4.2.2.4 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	-
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

Name	Type	Description	Default	Allowed vals.
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	-

3.4.2.2.5 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 carry the 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 level of the pre-processed cube(s) individually for each plane. Otherwise, the overscan subtraction step is skipped. (The calculated 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 will not 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 default 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.

3.4.2.2.6 Products:

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.

Name	Type	Description
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.

3.4.2.3 sph_zpl_master_dark

3.4.2.3.1 Purpose:

Create master dark, polarization modes.

3.4.2.3.2 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

3.4.2.3.3 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.

3.4.2.3.4 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

Name	Type	Description	Default	Allowed vals.
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	-

3.4.2.3.5 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 carry the 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 level of the pre-processed cube(s) individually for each plane. Otherwise, the overscan subtraction step is skipped. (The calculated 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 cameras are 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.

3.4.2.3.6 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.

Name	Type	Description
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.

3.4.2.4 sph_zpl_intensity_flat

3.4.2.4.1 Purpose:

Create intensity flat field, polarimetric modes.

3.4.2.4.2 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

3.4.2.4.3 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.

3.4.2.4.4 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	-

3.4.2.4.5 Description:

This recipe creates the master intensity flat field calibration frame for 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 files in 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 then applied 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 for all 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 data m_i,v_i.
4. The fit itself is performed either using a maximum likelyhood method or 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) can be 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 user parameters 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 above a 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 badpixels that are stored in the master flat field itself as produced by this recipe contain all the badpixels (for each sub-frames individually) at this point in the cascade. Pixels that were 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 the flat field recipe itself deemed to be bad. This does not necessarily include all the badpixels from the static badpixel input file.

3.4.2.4.6 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.

3.4.2.5 sph_zpl_polarization_flat

3.4.2.5.1 Purpose:

Create polarization flat field, polarimetric modes.

3.4.2.5.2 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

3.4.2.5.3 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.

3.4.2.5.4 Parameters:

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	-

Name	Type	Description	Default	Allowed vals.
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	-

3.4.2.5.5 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 flat-frames, which should carry the ZPL_POL_FLAT_PREPROC_CAM1 and/or tag ZPL_POL_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, and master 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 and then the Stokes parameters (I,P) are calculated. The output master polarization flat field is written out in the DOUBLE IMAGE (8 images) format specified as follows: - master intensity Stokes parameter image, badpixel-map, ncomb-map and rms-map; - master polarization Stokes 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.

3.4.2.5.6 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.

3.4.2.6 sph_zpl_modem_efficiency

3.4.2.6.1 Purpose:

Create modem efficiency, polarimetric modes.

3.4.2.6.2 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

3.4.2.6.3 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)

3.4.2.6.4 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	-

Name	Type	Description	Default	Allowed vals.
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	-
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

Name	Type	Description	Default	Allowed vals.
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	-

3.4.2.6.5 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, and 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. 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 Stokes 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 Stokes parameters (I,P) are calculated for both frames and both cameras in the form of DOUBLE IMAGE: - master intensity Stokes parameter image, badpixel-map, ncomb-map and rms-map; -

master polarization Stokes parameter image, badpixel-map, ncomb-map and rms-map. Then, the polarization flat field is applied to the [Qplus, Qminus] Stokes 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.

3.4.2.6.6 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.
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.

Name	Type	Description
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.

3.4.2.7 sph_zpl_star_center_pol

3.4.2.7.1 Purpose:

Determine the center of the star center frame, polarimetry modes.

3.4.2.7.2 Input frames:

Data Type (TAG)	Source	Optional	Min	Max
ZPL_ STAR_ CENTER_ POL_ RAW	Raw data	Yes	0	Any
ZPL_ STAR_ CENTER_ POL_ PREPROC	Calibration	Yes	0	Any
ZPL_ STAR_ CENTER_ POL_ PREPROC_ CAM1	Calibration	Yes	0	Any
ZPL_ STAR_ CENTER_ POL_ 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

3.4.2.7.3 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. SPH_COMMON_KEYWORD_DROT2_MODE string 0 0 0 De-rotator mode: ELEV(pupil stabilized), SKY(field stabilized)

3.4.2.7.4 Parameters:

Name	Type	Description	Default	Allowed vals.

Name	Type	Description	Default	Allowed vals.
zpl.star_ center.outfilename	string	The output filename for the star center product, camera-1/2. Please also see the esorex documentation for naming of output products.	zpl_ star_ center_ pol.fits	-
zpl.star_ center_ cam1.outfilename	string	The output filename for the star center product, camera-1. Please also see the esorex documentation for naming of output products.	zpl_ star_ center_ pol_ cam1.fits	-
zpl.star_ center_ cam2.outfilename	string	The output filename for the star center product, camera-2. Please also see the esorex documentation for naming of output products.	zpl_ star_ center_ pol_ cam2.fits	-
zpl.star_ center.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.star_ center.keep_ intermediate	bool	Flag to set if intermediate date must be saved, namely pre-processed and overscan pre-processed subtracted data (FALSE)	0	-
zpl.star_ center.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.star_ center.coll_ alg	int	Set the collapse algorithm. The available algorithms: 0 = Mean, 1 = Median. Default is 0 = Mean.	0	0,1,2
zpl.star_ center.filter_ radius	double	Filter radius for frame combination. A non zero value leads to suppression of high frequencies in the fourier domain before frame combination. 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.star_ center.sigma	double	The sigma threshold to use for source detections	10.0	-
zpl.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	-
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	-

3.4.2.7.5 Description:

The recipe produces combined frame of the star center calibration measurements in the polarization modes. The input frames might be either science polarimetric raw frames with the ZPL_STAR_CENTER_POL_RAW tag, or pre-processed science raw frames, which should carry the ZPL_STAR_CENTER_POL_PREPROC_CAM1 and/or ZPL_STAR_CENTER_POL_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 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 pre-processed frames are then calibrated by subtacting a corresponding master bias frame and a master dark frame, and dividing the results by a given intensity flat field frame. Then the Stokes 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. Then the calibrated frames are de-dithered (if needed) and saved as intermediate products (note: if the zpl.science_p1.save_interprod is set to TRUE, 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 calibrated and de-dithered frames are avareged using collapse mean algorithm. The combined start center frames 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. The combined star center image is then analysed using an aperture detection algorithm: - The aperture detection algorithm detects all connected regions of 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 coronograph is out and no waffle stars are formed). The coordinates of the detected frame centers are added as the keywords in the header of the output double images for each camera: - DRS ZPL STAR CENTER IFRAME XCOORD; - DRS ZPL STAR CENTER IFRAME YCOORD; - DRS ZPL STAR CENTER PFRAME XCOORD; - DRS ZPL STAR CENTER PFRAME YCOORD. This star center calibrated products for each camera should be used in the science polarimetric recipes.

3.4.2.7.6 Products:

Name	Type	Description

Name	Type	Description
ZPL_ STAR_ CENTER_ POL	FITS[Im(8)]	The final combined star center calibration frame [I, P] is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - combined intensity image of the star center calibration measurement with corresponding badpixel-map, ncomb-map and rms-map; - combined polarization image of the star center calibration measurement with corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ STAR_ CENTER_ POL_ CAM1	FITS[Im(8)]	The final combined star center calibration frame [I, P] for the camera-1 is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - combined intensity image of the star center calibration measurement with corresponding badpixel-map, ncomb-map and rms-map; - combined polarization image of the star center calibration measurement with corresponding badpixel-map, ncomb-map and rms-map;
ZPL_ STAR_ CENTER_ POL_ CAM2	FITS[Im(8)]	The final combined star center calibration frame [I, P] for the camera-2 is of the DOUBLE IMAGE format. This frame contains 8 image extensions: - combined intensity image of the star center calibration measurement with corresponding badpixel-map, ncomb-map and rms-map; - combined polarization image of the star center calibration measurement with corresponding badpixel-map, ncomb-map and rms-map;

3.4.2.8 sph_zpl_science_p1

3.4.2.8.1 Purpose:

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

3.4.2.8.2 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_STAR_CENTER_POL_CAM1	Calibration	Yes	0	1
ZPL_STAR_CENTER_POL_CAM2	Calibration	Yes	0	1
ZPL_STAR_CENTER_POL	Calibration	Yes	0	1
ZPL_FIELD_CENTER_TABLE	Calibration	Yes	0	Any
ZPL_POLHIGH_STAR_TABLE	Calibration	Yes	0	1
ZPL_FILTER_TABLE	Calibration	Yes	0	1
ZPL_POL_CORRECT_TABLE	Calibration	Yes	0	1

3.4.2.8.3 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)

3.4.2.8.4 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	-
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	-

Name	Type	Description	Default	Allowed vals.
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	-
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	-

Name	Type	Description	Default	Allowed vals.
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 data must be saved, namely pre-processed and overscan pre-processed subtracted data (FALSE) Note that this setting this parameter to TRUE will use a very large amount of disk space	0	-
zpl.science_ p1.save_ interprod	bool	Flag to set if the field center table, plus the final calibrated cube without rotation, must be saved as intermediate product (FALSE) Note that this parameter must be only applied for the offline pipeline	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 frame combination. A non zero value leads to suppression of high frequencies in the fourier domain before frame combination. 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_ p1.star_ center_ iframe	bool	Flag to set if only the center coordinates of the iframe from the star center calibration frame should be used as a center coordinates to de-rotate iframe and pframe (TRUE)	1	-
zpl.science_ p1.center_ xoffset_ cam1	double	X-offset from the center of the image for cam1	0.0	-512.0-512.0
zpl.science_ p1.center_ yoffset_ cam1	double	Y-offset from the center of the image for cam1	0.0	-512.0-512.0
zpl.science_ p1.center_ xoffset_ cam2	double	X-offset from the center of the image for cam2	0.0	-512.0-512.0
zpl.science_ p1.center_ yoffset_ cam2	double	Y-offset from the center of the image for cam2	0.0	-512.0-512.0
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	-

3.4.2.8.5 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 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 Stokes 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 Stokes 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 TRUE, the recipe will also save the so called field center table which contains the the calculated center positions and parallactical angles (to be more specific: an angle to be used for the de-rotation) for each plane of the pre-processed fits cube(s)). Note: the calibration frames with SPH_ZPL_TAG_STAR_CENTER_POL_CALIB_CAM1(CAM2) tags provide the center coordinates to rotate around. If these calibrations are not presented the center of the frames will be used (normally, xc=yc=512 pixel). 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.

3.4.2.8.6 Products:

Name	Type	Description

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;
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;

Name	Type	Description
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;
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;

Name	Type	Description
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;
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;

Name	Type	Description
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;
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;

Name	Type	Description
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;
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;

3.4.2.9 sph_zpl_science_p23

3.4.2.9.1 Purpose:

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

3.4.2.9.2 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_ SCIENCE_ P23_ PREPROC	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_ STAR_ CENTER_ POL_ CAM1	Calibration	Yes	0	1
ZPL_ STAR_ CENTER_ POL_ CAM2	Calibration	Yes	0	1
ZPL_ STAR_ CENTER_ POL	Calibration	Yes	0	1
ZPL_ FIELD_ CENTER_ TABLE	Calibration	Yes	0	Any

3.4.2.9.3 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)

3.4.2.9.4 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	-
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	-

Name	Type	Description	Default	Allowed vals.
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.keep_ intermediate	bool	Flag to set if intermediate data must be saved, namely pre-processed and overscan pre-processed subtracted data (FALSE) Note that this setting this parameter to TRUE will use a very large amount of disk space	0	-
zpl.science_ p23.save_ interprod	bool	Flag to set if the field center table, plus the final calibrated cube without rotation, must be saved as intermediate product (FALSE) Note that this parameter must be only applied for the offline pipeline	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
zpl.science_ p23.filter_ radius	double	Filter radius for frame combination. A non zero value leads to suppression of high frequencies in the fourier domain before frame combination. 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_ p23.star_ center_ iframe	bool	Flag to set if only the center coordinates of the iframe from the star center calibration frame should be used as a center coordinates to de-rotate iframe and pframe (TRUE)	1	-
zpl.science_ p23.center_ xoffset_ cam1	double	X-offset from the center of the image for cam1	0.0	-512.0-512.0
zpl.science_ p23.center_ yoffset_ cam1	double	Y-offset from the center of the image for cam1	0.0	-512.0-512.0
zpl.science_ p23.center_ xoffset_ cam2	double	X-offset from the center of the image for cam2	0.0	-512.0-512.0
zpl.science_ p23.center_ yoffset_ cam2	double	Y-offset from the center of the image for cam2	0.0	-512.0-512.0
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	-

3.4.2.9.5 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 Stokes 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 Stokes 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.

3.4.2.9.6 Products:

Name	Type	Description

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;
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;

Name	Type	Description
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;
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;

Name	Type	Description
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;
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;

3.4.3 ZIMPOL Workflow Summary

The ZIMPOL imaging workflow is summarized in Fig.3.6, the polarimetric workflow in Fig.3.7.

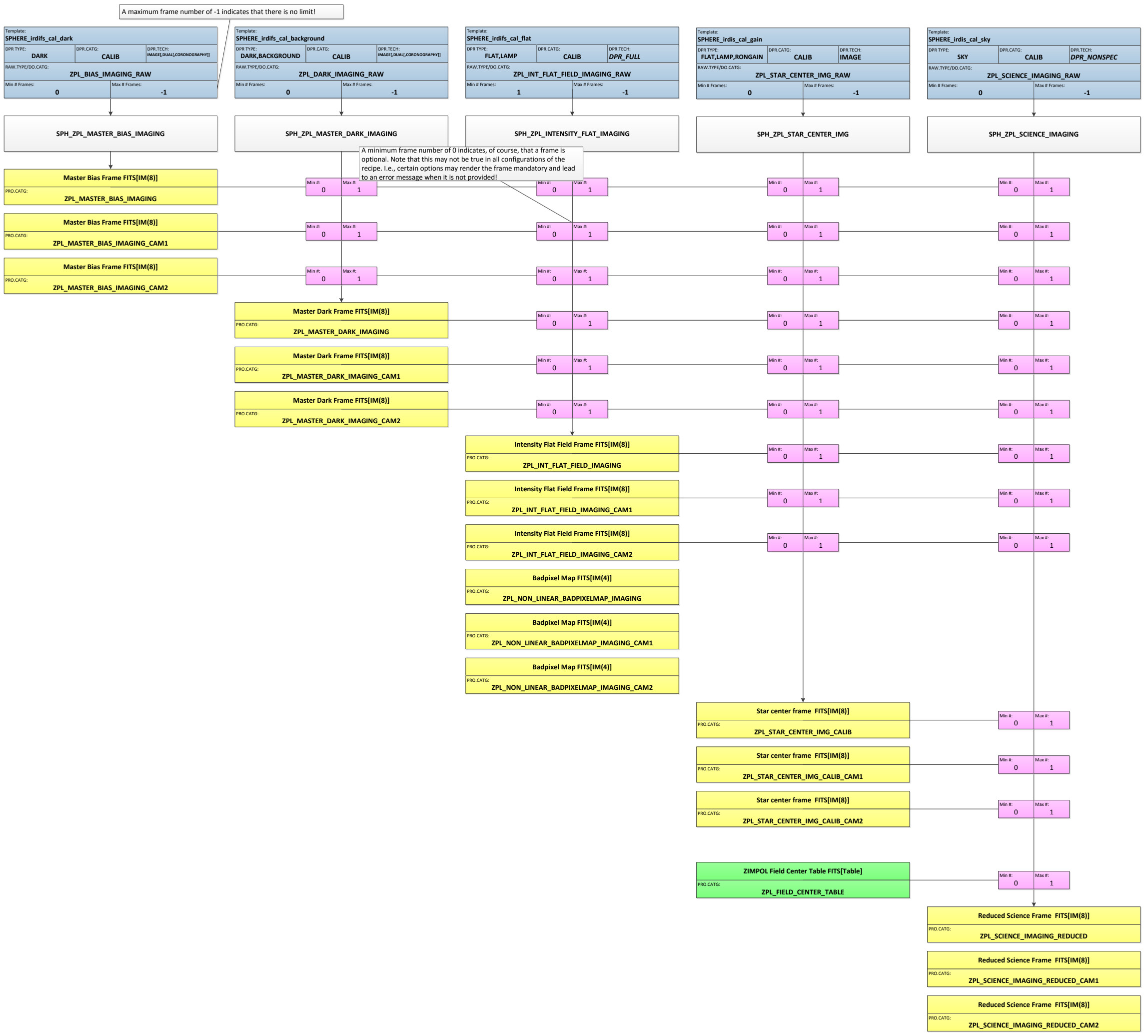

Figure 3.6: ZIMPOL Imaging Workflow

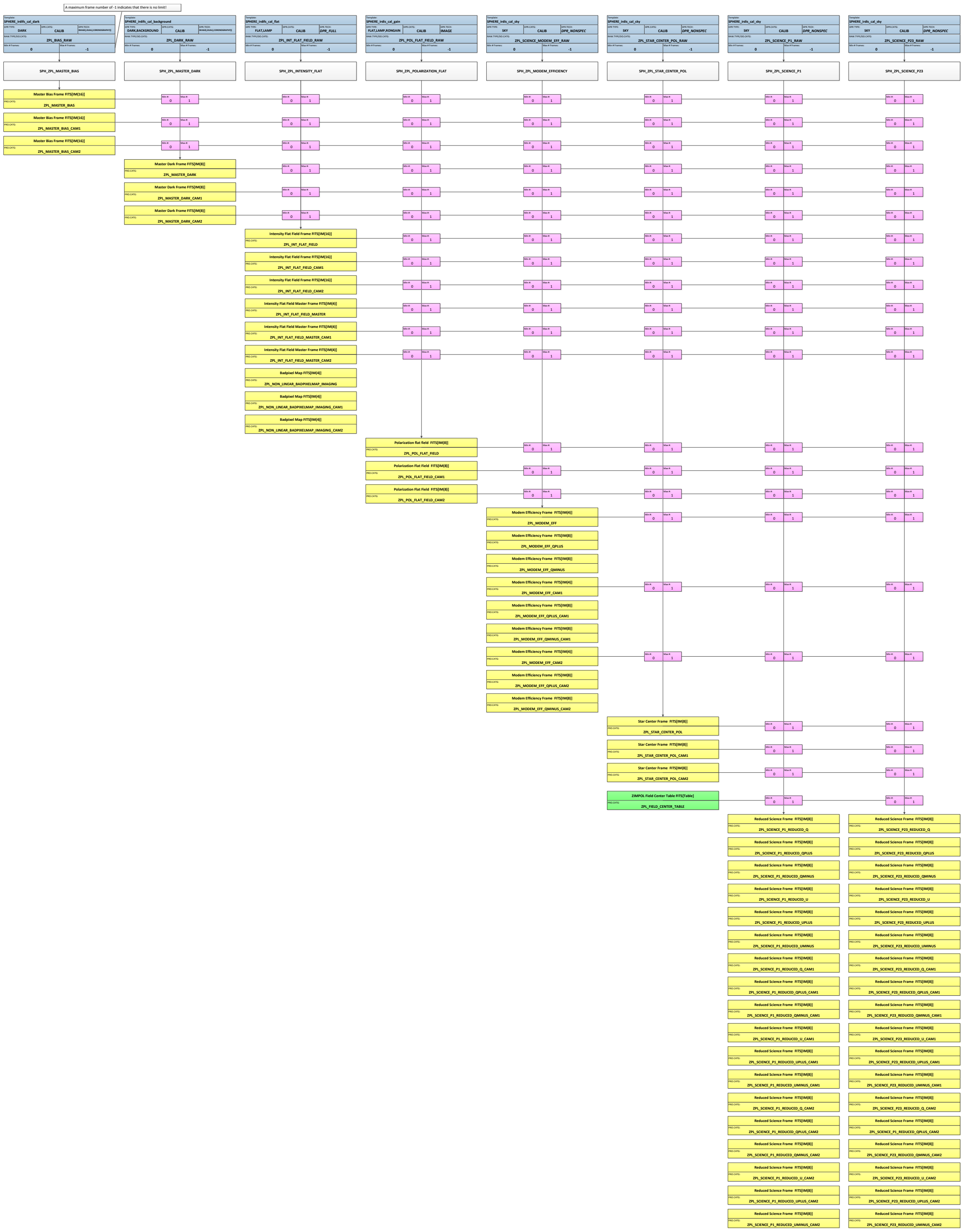

Figure 3.7: ZIMPOL Polarimetry Workflow

Chapter 4

Instrument Data Description

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

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

General Data Layout A raw SPHERE file always has the images stored in the primary FITS dataunit.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 listedfor each recipe.

Imaging Frames The raw imaging frames for IFS and IRDIS all contain in total 2048x2048pixels.For IRDIS all raw frames that are obtained with calibrations that illuminatethe detector through the IRDIS optical path, only an area of 2048x1024pixels is used.This again is split in two parts for classical imaging and DBI modes.

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

Coordinates reported (or used as input) by the SPHERE pipeline always referto 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 rightpixel.The midpoint of the lower left pixel has the coordinate (0.5,0.5), and theimage midpoint (N/2,M/2).

4.1 ZIMPOL Data

4.1.1 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"
```

4.2 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 locatedin sphere-0.14.1/spherec/cal The current static calibration data availablefor SPHERE are the IRDIS and IFS instrument models.

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 instrumentcoordinates 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 4.1: IFS lenslet model parameters.Parameters marked with a (*) are derived from other quanitites and cannot be changed directly, parameters mared with a § are fitted for in thespectra positions recipe.

```
> 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:
```

4.2.1 IFS lenslet model

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

4.2.1.1 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 followingway:

```
[ 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 (spectralength, minimum and maximum wavelength) are illustrated in Fig.4.1.A cross section of a typical spectrum with a broad band lamp is shown atthe top of the figure.Below, the diagram illustrates the meaning of the principal parametersand 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 PIXLEN and ESO DRS IFS SPEC PIX WIDTH parameters.The exact position of the spectrum on the detector is determined by thehexagonal 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 IFSSCALE 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 fullyinside 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 alwaysonly be 38 pixels long.The minimum and maximum wavelength then refer to the midpoints of the firstand 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 theactual 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: $\Delta\lambda = (\lambda_{max} - \lambda_{min}) / (L_{extract} - 1)$.

4.2.2 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 containsinformation on the detector regions corresponding to the different opticalpaths (left and right paths).It is automatically created in the instrument flat recipe and the modelinformation 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 IRDISinstrument model parameters, like filter parameters, etc.In the current version, the lenslet model can be provided most easily asa simple ASCII text file which is written in the “ini”style format of “KEY = VALUE”pairs on each line.

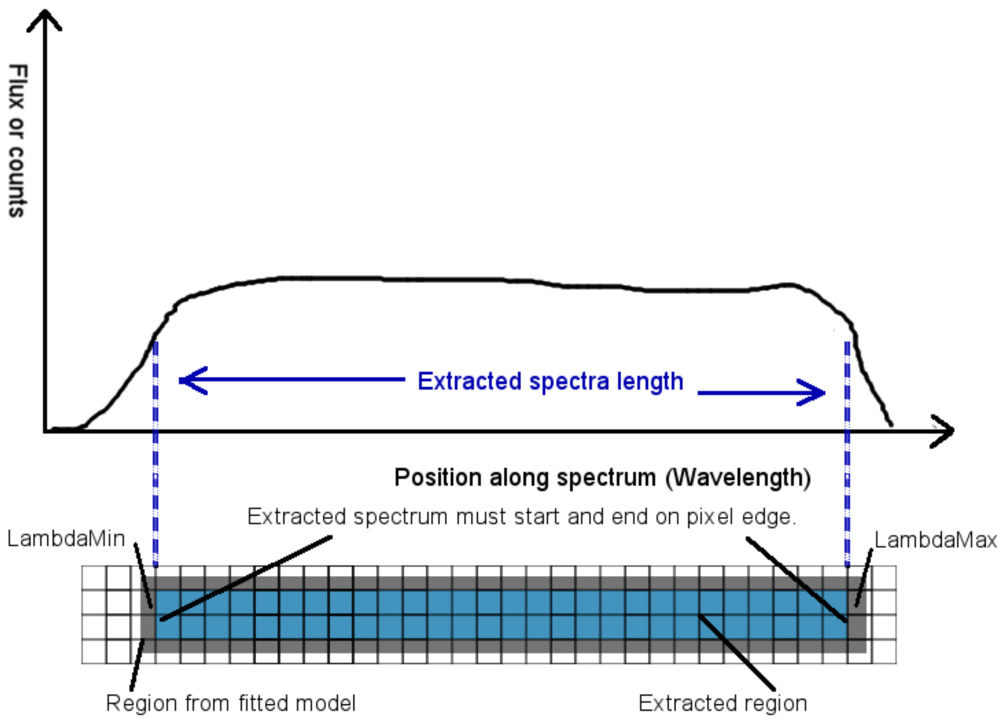

Figure 4.1: The spectra model and spectra extraction for IFS

4.3 Data Reduction Pipeline Data Products Format

4.3.1 Calibration Products Data Representation

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

4.3.1.1 The SPHERE "master frame"

The most simple data product produced by the SPHERE pipeline consists ofa 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 andIRDIS 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

4.3.1.2 Seeing double: The SPHERE double image

For all instruments several calibration (and also raw data) consist of twoassociated images.This is specifically true for IRDIS, which uses a double optical path closeto the detector, and ZIMPOL which stores two interlaced separate imagesin one readout detector frame using pixel-shifting.Also for IFS some information, like the distortion vector of the lensletarray 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 thesph_ird_science_dpi recipe, the “A”image represents the intensity image, I, and image “B”the polarisation, P.

4.3.1.3 The SPHERE quad image

Some of ZIMPOL instrument calibration output product consists of 4 associatedimages.This is the consequence of the fact that one zimpol exposure contains twointerlaced 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 ZIMPOLmaster bias and master dark.

Chapter 5

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 reductionprocess.

5.1 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 intocounts including all instrumental effects.Here the physical (or "sky") coordinates α,$\beta$are transformed onto the detector pixels x, ythrough 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 betweenlenslet array and detector, $\Delta x$and $\Delta y$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 instrumentaleffects.We write the λdependence here as a reminder that this is a 2D representation of a wavelengthcube.

From the entrance into the telescope the scientific signal is affected byseveral components in an adverse manner, and all these effects have tobe removed by the data reduction process in order to achieve maximal scientificoutput.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, reflectingthe physical layout of the detecting system, which consists of severalcomponents 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 inmind.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 individualcomponent in a calibration procedure which is executed in a separate stepto the science observation, either at various times during the observationnight, during the preceding day or only at specific times throughout theyear.

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

Figure 5.1: Schematic representation of the hardware components for IFS from a DRH pointof view.The signal is affected by various hardware components which are calibratedout 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 followingequation:

$$I(x, y) = G \times \{DC(x, y) \times \Delta t + B(x, y) + DF(x, y, \lambda) \times IF(x, y, \Delta x, \Delta y, \lambda) \times S(x, y; \lambda)\} + RON, \tag{5.1}$$

where G is the total gain, DC the dark current, B the bias, DF the detectorflat response, IF the instrument flat response, RON the readout noise,$\Delta x$the detector dither offset in x, $\Delta y$the detector offset in y.The exposure time, Δthas 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 measureddirectly for IFS and IRDIS but has to be inferred.

All the functions for the system components, DC, B, DF, IF and TF are writtenin detector pixel coordinates, even if the corresponding calibrations maybe detector position independent.For example, the instrument flat field is the effect of the lenslet arrayon the signal, which is a function of lenslet and wavelength but is independenton the detector position.The signal as received in detector pixel coordinates, however, is dependenton the detector offset simply due to a shift in coordinate system.In this sense the functions for the system components defined in the equationabove rather represent the detected signal on the detector if all othercontributions are zero.The response functions of the detector, the instrument and the telescopeare assumed to be linear in the signal S.Linearity of these components, for signals in unsaturated regimes, is partof the SPHERE hardware requirement specification and the linearity assumptionis therefore in general justified.An exception are image ghosts (due to optical reflections) and persistenceeffects.

5.2 Signal propagation reversal

Given the above signal response equation, the inverse can be formulatedto 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:

$$S(x, y; \lambda) = \frac{I(x, y)/G - DC(x, y) \times \Delta t - B(x, y)}{DF(x, y, \lambda) \times IF(x, y, \Delta x, \Delta y, \lambda) \times TF(x, y, \Delta x, \Delta y, \lambda)}. \tag{5.2}$$

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

5.3 Signal propagation for bias and dark calibrations

The signal propagation for bias and dark calibration is very simple sincethe 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 biasand 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 dependson the readout mode, a bias+dark measurement is required for each exposuretime and readout mode used in any observation which is to be processed.This is generally true for all detector effects and will be neglected inthe further treatment in this chapter (the only consequence is that everymeasurement is performed for each possible combination of exposure timeand 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 fordifferent times Δtthereby 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 isnot possible for the infrared detector and so the bias can not be measuredusing 0s exposures as for optical CCDs.

5.3.1 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 thatshould 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 pixelsto a zero input stimulus.This is dependent on integration time as well as read out mode and mayvary on relatively short timescales.To keep with the terminology of the calibration plan we shall continueto refer to this calibration as the "dark"calibration also for the infrared detectors of IFS and IRDIS.

5.4 Signal propagation for the detector flat field

In this case, the detector is illuminated with a uniform lamp of a givenwavelength, 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 telescopeare 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, andexploiting 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 wavelengthis measured by taking an exposure of time $\Delta t$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 5.2is $DF(x, y, \lambda)$, 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 bandwidthand 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, itis 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 fitfunction 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 calibrationlamps used, and how well the wavelength dependence of the pixel responsecan be described by the fitting function chosen.

5.5 Signal propagation for the instrument flat field

5.5.1 Instrument flat field for IFS

For the instrument flat field for the IFS, the set-up is similar to thedetector flat field, except that the lenslet array and some related opticalcomponents are in the light path.The equation for the signal propagation, eq.5.1becomes:

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

where $L(\lambda)$is the spectral energy distribution of the calibration lamp and we haveassumed that the mean of the readout noise is zero.Now, the calibration measurement is the same as that for the detector flatfield, 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, detectorflats are independent of the detector position.However, the pixel associated is detector position dependent, and so, inorder to measure $IF(x, y, \lambda)$the pixel positions have to be remapped through the pixel description tablebefore a detector flat is divided out.The main purpose of the sph_ifs_instrument_flat recipe described lateris 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 detectorposition as long as detector flat fields taken at the same detector positionare divided out.Again the limited availability of calibration lamps means that the wavelengthdependence will be estimated by functional fits, $f_{x,y}(\lambda)$, to a series of measurements at the different calibration wavelengths,such that

$$DF(x, y, \lambda) \times IF(x, y, \Delta x, \Delta y, \lambda) = f_{x,y,\Delta x,\Delta y}(\lambda). \quad (5.3)$$

Recipes that need to correct for the instrument flat field use a set ofmaster input detector flat fields in combination with the detector positionindependent 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 aparameter to the recipe plugin.We should note here, that the λdependence of the lenslet response is expected to be small – and it mayin principle be possible to simplify the data reduction process in thiscase.

For IFS the quantity on the left hand side of equation5.3multiplied by the wavelength association mask as obtained during the wavelengthcalibration (described further below)

$$SF(x,y) \equiv \int DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda) \times \delta(\lambda - \lambda_{x,y})\, d\lambda \tag{5.4}$$

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 wavelengthcalibration.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 enoughto measure the quantity SF(x,y) in one single calibration since the differentquantities entering S(x,y) vary on different timescales.Therefore the super fat is reconstructed from separate master calibrationfiles of DF, IF and the wavelength calibration within all science observationand calibration data reductions.Also note that SF(x,y) is dither position dependent.

5.5.2 Instrument flat for IRDIS

Fir IRDIS there is no detector flat field, and there exists only the instrumentflat 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

$$FF_{imaging}(x,y;F) \equiv \int DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda) \times F(\lambda)\, d\lambda, \tag{5.5}$$

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 arenot 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 wayto IFS as

$$FF_{spec}(x,y;F) \equiv \int DF(x,y,\lambda) \times IF(x,y,\Delta x,\Delta y,\lambda) \times F(\lambda) \times \delta(\lambda - \lambda_{x,y})\, d\lambda. \tag{5.6}$$

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

5.6 Finding spectral regions in IFS

For the IFU capabilities of the IFS it is necessary to identify all theregions where spectra fall onto the detector in an automatic way.In principle this needs to be done for every possible detector positionin a separate calibration step implemented as the sph_ifs_spectra_positionsin the data reduction library.However, the creation of detector position dependent PDTs is done purelyin 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 positionand 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 abovea certain threshold value.

4. Assign a label for each connected region.

5. Associate these regions from the regions as expected from a model of thelenslet array.Regions that are either associated to two different lenslet IDs or thathave 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 spectralregions 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 requiringa flat spectra response, that is, the signal along a spectrum must be highand 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 stronglywavelength dependent sensitivity the spectra will not be flat on the detectorand in some regions the detector sensitivity may be so low that the contrastis not high enough.In addition this procedure does not take account of the fact that the boundaryof spectra do not fall exactly in between pixels; some pixels will be illumination partly by a spectrum, further reducing the contrast with un-illuminatedregions.All this means that the performance of this procedure depends rather criticallyon the choice of the threshold parameter.

An alternative approach, used in the current SPHERE pipeline, uses a modelfunction of the spectra locations to improve on the simple thresholdingapproach.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 spectralregions 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 in4.2.1.For all the details on the spectra positions procedure please see the recipedescription in **??**.

5.7 The IFS wavelength cube and IFS wavelength calibrations

5.7.1 The wavelength cube

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

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

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 betweenlenslet array and detector and are determined during the spectra positionsprocedures 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 thepixel coordinates of the known line centers of the spectrum.For every lenslet spectrum a table associating the line wavelengths withpixels information is constructed and a fitting/interpolation procedureis used to associate wavelengths for all pixels in between.In this way every pixel will be associated with a lenslet (i.e.$\alpha$and βcoordinates) and a wavelength.Since the procedure makes use of the same instrument set-up as used forthe 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 interestedin 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 lensletarray.Using the reversed propagation equation 5.2, 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.

5.7.2 Wavelength calibration

At the wavelength calibration stage, the information of the pixel to wavelengthassociations is not yet available (that is rather the result or purposeof the wavelength calibration) the flat fielding has to be performed hereusing a IFS flat field frame which has not been divided by the detectorflat, 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 flatfield defined in 5.3.So, as a first step, after dark subtraction, the raw wavelength calibrationframes are divided by the super flat field (as measured directly in thesph_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 calibrationlamp.The calibration lamp produces very sharp, monochromatic lines.The line profile as observed on the detector are a convolution of the intrinsicline profile, negligible for the calibration lamp lines used, and the instrumental profile (spectrograph resolution).Therefore, the expected line width for these new calibration hardware willbe 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 lineshave to be determined by pre-selecting the regions close to the expectedpositions of the lines to avoid the fitting to be performed on some ofthese fringes (present as local maxima).Alternatively, the data extracted from the spectra region is first passedthrough 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, wherespectra resolution is low, but for IRDIS MRS spectroscopy line fringinghas to be taken into account.Only the first, simplest method is currently implemented.The peak position for each line is determined by calculating the weightedmean position for a window of a few pixels size around the expected lineposition.The expected line position is taken from an input model of the spectrapositions and the dispersion.Since this input can also be another wavelength calibration product, thewavelength 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 wavelengthinformation.If $P > 1$the second derivative is used to estimate the dispersion.[1]

5.8 Further spectral and flux calibrations

The IFS uses IFU capabilities to create a wavelength data cube as the mainscience 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 thesource.However, the observed spectrum is, just like the detector image, affectedby several unwanted effects: the atmosphere as well as telescope subsystemintroduce wavelength dependent effects.The observed spectrum is given by:

$$F_{obs}(\lambda; z, r, \theta) = F_{real}(\lambda) \times A_{atm}(\lambda; z) \times A_{corono}(\lambda; r, \theta) \times A_{tel}(\lambda; r, \theta) + T_{atm}(\lambda; z), \tag{5.7}$$

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 thetelescope, respectively, $T_{atm}(\lambda; z)$is the atmospheric transmission at wavelength $\lambda$and $F_{real}(\lambda)$is the true scientific signal to detect, which needs to be reconstructedin the calibration procedure.In order to be able to do this, without the need to obtain calibrationframes for every science exposure, it is necessary to model the variousinstrumental and atmospheric effects individually and measure the relativecontributions and model parameters at regular intervals.To this end, the various effects can be disentangles making use of thedifferent dependencies: the atmospheric effects depend on airmass, $z$but are independent of source location on the detector, whereas telescopeand coronagraph affect the signal dependent on the source position withinthe frame but are independent of air mass.The various components are modeled and calibrated as follows.

5.8.1 Atmospheric absorption

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

5.8.2 Coronagraph effects

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

In the rare cases where a removal of the coronagraph effect is explicitlyrequired, exposures can be taken with and without coronagraph and the ratioof the resulting spectra gives directly the

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

coronagraphic effect (exceptfor an unknown normalization constant).However, the fact that measurements are necessary at different points inthe field makes this rather problematic if no good model of the coronagraphiceffect 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 determinedin the first weeks after or during commissioning.

5.8.3 Instrumental background and sky background

In 5.1no additive effects are included except those arising from the detector.However, there are signal contributions from both the sky and the instrumentitself which need to be removed in order to obtain the true science signal.Since these are additive effects, they need to be subtracted out afterthe reverse propagation equation above, 5.2has 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/removalof the sky and instrument background follows a similar procedure to thedark calibration, in the sense that they are subtracted from the inputframes.Also note that all contributions have a wavelength dependence.

For IFS the contributions of the sky and instrument background are expectedto be small and unimportant for the main science objective: the detectionand 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 sciencegoal of the observations.Therefore, even though recipes are included in the pipeline to calibratethese effects, their applicability will be limited.

5.8.4 Flux normalization calibration

Since the above calibrations are all performed using ratios, it is not possibleto determine in this way the absolute flux.In order to do this, one needs to observe a known source and compare totalreceived 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.

5.9 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 isnecessary to repeat calibration procedures at time intervals which areless than the stability time-scale for the required accuracy.For example, the detector flat field is stable to within 0.1% only for aboutone hour.This means that the recipes that need to correct for effects that involvethe $DF(x, y, \lambda)$term need to use calibration measurements of this quantity that are maximallyone hour old.An alternative in such cases it to use monitoring measurements to constructa model of the time behaviour of the relevant subsystems.In the following table, we list approximate dependencies of the calibrationterms:

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 usedfor the calibrations have the required “freshness”– the pipelines will make no checks for that.Also note that some calibration procedures measure quantities that arecombinations of terms with different time variability.In these cases, the acceptable time-scale is given by the smallest acceptabletime-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 sucha 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 inwavelength 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 wavelengthdependence, $f(\lambda)$is almost constant over time, it is possible to only perform measurementsof $D(x,y)$frequently, which requires taking calibration data with only a single lamp.This is the approach taken in SPHERE.

5.10 Clean Mean Algorithm: Basic Frame Combination with outlier rejection

Please note the the spatial derivative pixel rejection described below hasnot been implemented in SPHERE.

The clean mean algorithm is used to average frames taking into account thepossibility 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 individualbad or outlying pixels at the same time as keeping the maximum amount ofinformation.

We use iterative clean mean with sigma computation (following that describedin ESO’s SINFONI Pipeline User Manual) as our baseline frame combinationmethod.For this process, the user sets minimum and maximum allowable intensityvalues.For a stack of frames, values inside this range are then used to determinean 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 achievingsuperior outlier rejection, compared to the clean mean method.This alternative procedure (presented in [RD2]) uses an iterative median/meancombination outlier rejection strategy that takes advantage of the spatialderivative of an image to better deal with variations from a changing PSFshape 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 valuesbeing wrongly interpreted as outlier pixels.The spatial derivative method should be effective at dealing with suchphenomena, as demonstrated in data reduction for Spitzer IRAC (see [RD3])and HST ([RD4]).

The code operates by first conducting a biased-median-combination of inputframes 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 sourceslike cosmic ray effects.From this *best estimate image*, *BEI,* a *spatial derivative array, SDA*, may be calculated using the following equation.

$$SDA(x, y) = maxabs(BEI(x, y)\check{} [BEI(x - 1, y), BEI(x + 1, y), BEI(x, y + 1), BEI(x, y - 1)]) \quad (5.8)$$

Going back to the original input frames, we remove any pixel that differsin 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 followingcondition.

$$|original - frame(x, y)\check{} BEI(x, y)| > k \times SDA(x, y) \quad (5.9)$$

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

5.11 Detector pixel linearity

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

The linearity measurement is used in all recipes where the detector responseis assumed to be linear as part of the algorithm.This is the case for all recipes that divide out the detector flat fieldfor example, since the exact detector response is a function of input signal,and extrapolation to the actual input signal from the available detectorflat calibration frames is needed (the value $DF(x, y, \lambda)$in the equation 5.1is just this linear detector coefficient).

Recipes that need to correct for the detector flat field actually use thedetector pixel linearity for the flat fielding, since this gives the correctionas 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 withlinearity 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 excludedfrom a frame combination procedure.The identification of bad pixels using this method is possible both fordynamic and static bad pixel identification – but its main use is to identifystatic 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 dynamicbad pixel identification works as required and that the dynamic bad pixelidentification routine are not affecting further data calibration and reductions adversely.

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

5.12 Bad Pixel Identification

Bad pixel identification in SPHERE happens on several levels and in severalways.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 thename 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 combinea set of raw frames into a smaller set of output frames, e.g.the CLEAN MEAN algorithm described in 5.10.

5.12.1 Static Bad Pixel Identification

The identification of static bad (dead or hot) pixels in the data reductionoccurs usually during the master dark calibration recipe.When the master dark frames are created, a static bad pixel map is createdin 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 exposuretime and readout.
2. The individual frames are combined using the clean mean method, effectivelyremoving 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 notrandom chance events.However, note that a reliable identification of truly static pixels requiresthat 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 linearitybehaviour to determine bad pixels.During the sph_ifs_detector_flat fielding recipe, a detector linearitymap is created (see 5.11).This map gives, for every pixel, a measure of the linearity (goodness offit 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 linearityto accept/reject pixels.The detector flat recipe itself creates a static bad pixel map in thisway, 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_darkrecipe, is used primarily for monitoring purposes and to validate the staticbad pixels identified in the sph_ifs_master_detector_flat field recipe.For that purpose the detector flat field recipe outputs a quality controlparameter that measures the number of pixels that have been identifiedin one static bad pixel map, but not the other.A large number here usually means that the detector linearity performancehas degraded.

5.12.2 Dynamic Bad Pixel Identification

Apart from static bad pixels due to faults in the detector, there are alsodynamic bad pixels that are created by transient effects: most notablyby cosmic rays.These are identified whenever raw frames are combined.Each frame combination routine, like the clean mean algorithm above, needsto reject pixels that are deemed as bad – in the case of the clean meanalgorithm due to their value away from the mean value.Since bad pixels due to cosmic rays can affect neighboring pixels as wellas 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 subsequentframe.This happens, for pixels that are sufficiently outlying, automaticallyin the clean mean algorithm.

5.12.3 Bad pixel treatment in the IRDIS and ZIMPOL science recipes

The science reduction recipes for IRDIS and ZIMPOL use the bad pixel mapsfrom the dark and the flat field recipes to set the bad pixels on the resultingreduced science images.The bad pixels set in the science frames is always the union of all staticbad pixel maps.Both IRDIS and ZIMPOL use geometric transformations as part of the usualscience data reduction process (e.g.for differential imaging or to process dithered frames) and the bad pixelmaps are transformed along with the image.The algorithm for transforming the bad pixel maps is purely geometric inany case even when the image data is transformed using an FFT.Due to serious artefacts that would arise if an FFT is performed on datacontaining many discontinuities arising from bad pixels, the science recipesinterpolate the bad pixels before any transformation on the image data.The (also transformed) bad pixel map however is maintained and used tocalculate a final bad pixel map on the fully reduced and combined scienceimage that then shows as bad all pixels that had no valid input pixel information.

5.13 Field Center

For all SPHERE instruments accurate determination of the field center isimportant.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 rawframes 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 satisfactoryaccuracy <0.5 mas)
- the DTTS calibration (CPI-TEC-01) outputs the position on the IRDIS detectorof the coronagraph (for its internal use)

Currently there are several different field center calibration strategiesconsidered:

1. Calibration within science data reduction recipes: here one uses the scienceraw frames themselves and the fact that the coronagraph center can be easilydetermined by determining the center of the region masked by the coronagraph.The AO (DTTS loop) then ensures that this coronagraph center is also therotation center.For frames taken without coronagraph the star center itself, which is easilydetermined by finding the peak and Gaussian fitting.
2. Calibration of the center in a dedicated recipe but using the input rawscience observation frames.For IRDIS the star center calibration which uses 4 secondary diffractionpeaks 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 calibrationwhich determines the center of the coronagraph for all possible instrumentset-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 referencefield center to use for frame combination in science data reduction recipes.

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

5.14 Frame combination: de-shifting and de-rotating

Frame combination is one of the most crucial steps in the SPHERE data reductionsince accurate frame combination including de-shifting and de-rotationallows 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 theother hand and to be flexible and modular on the other.This is realised by allowing the choice of frame-combination to be givenas an input parameter to recipes, along with all relevant parameters neededfor 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 isthe FFT.The exact FFT implementation is a separate module and de-coupled from theactual 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 isnecessary).

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

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

5.14.1 GIMROS - Generic IMage ROtation and Scaling

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

5.14.1.1 The concept behind GIMROS

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

5.14.1.2 Transforming an image with GIMROS

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

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

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

It is obvious that this technique is flux conserving as long as both imagesare completely and continuously covered with non-overlapping pixels.Gaps between pixels, particularly in the output image, may of course leadto 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 badpixel maps on the input side

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

```
1.calculate all centre points of all (transformed) input and output polygons
2.find the longest distance between centres and any edge point throughoutthe input an
3.sort the pixel polygons by the distance of their centre point to the centrepoint of
4.the potentially overlapping pixels are the ones where the distance is below2*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.

5.15 Creating wavelength cubes for IFS

One of the last reduction steps in producing calibrated science frames forIFS is to construct a wavelength data cube, $S(\alpha, \beta, \lambda)$.As described before, this is achieved using pixel-to-lenslet associationtables.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 rectangularstructure.This means that every spatial x, yposition in the wavelength cube has a non-trivial associated sky positionα,β.How this geometric transformation is performed depends on the level ofscience reduction.For the basic reduction there is two possible outputs: a result on a hexagonalgrid 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 performeddepends 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 detectorimages.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 ditheringposition.For each dithering position used in the observations to combine, a newPDT has to be constructed.This is done purely in a software manner – calculating a new PDT for theoffset 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 4.2.1.Extracting the spectra and moving to the lenslet view removes any ditheringdependence 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 foreach lenslet) the result is saved as a wavelength cube.For this, the hexagonal structure is interpolated onto a hexagonal grid.We have currently implemented a method based on GIMROS to project hexagonsonto quadratic pixels.

5.16 Distortion map

The distortion map needs to be measured for IFS and IRDIS.For ZIMPOL no distortion map measurement is currently foreseen, but itmay 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

$$S(x, y) = S'(x + \Delta_x(x, y), y + \Delta_y(x, y)) \tag{5.10}$$

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

In both cases, the distortion is measured using a grid of artificial sourceswith known positions.The grid positions is used as an input to the distortion map recipe whichdetects the actual observed sources and measures their displacement withrespect 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.

5.17 Astrometry and plate scale solution

The field of view for all SPHERE instruments is very small and astrometryfor SPHERE is a two parameter problem: the rotation angle relative to thenorth 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 needsto be observed.The relevant recipes reduced the raw observation frames and automaticallydetect the central star and the companion.Together with the user input parameters of angle and separation the angleto north and plate scale are derived.

5.18 ZIMPOL measurements

5.18.1 Definitions of terms used for ZIMPOL measurements

In this document and in the SPHERE ZIMPOL calibration plan (RD1), severalexpressions are used for describing specific parts of a ZIMPOL measurement.The different expressions aim to better distinguish between the entitiesthey describe and are chosen according to ESO definitions described inAD1.

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 storageof 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 opaquestripe 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 microlensesand exposed pixels.In imaging mode I1, where no demodulation takes place, the intensity imageis stored only in the *sub-frame*i^{A}while the other *sub-frame*i^{B}remains empty.

5.18.2 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-phasemode measurements.Each frame (number $k = 1...2{\cdot}\mathrm{NDIT}$) contains two sub-frames, $i_k^{\mathrm{A}}$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 thatthese two sub-frames represent an intensity image of two opposite Stokespolarization 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 cyclebetween each consecutive pair of frames, effectively exchanging the assignments(Q_+stored in i_{k+1}^{B}, Q_-in i_{k+1}^{A}).In Figure 5.2a graphical explanation is given.

When a cycle of NDIT frames is taken, the half-wave plate HWP2 is rotatedby 45°("HWP2 flip"), and the cycle of NDIT frames is repeated for the secondHWP2 position.When the observations for both HWP2 positions are finished, an optionaldithering is performed over NDITHER positions.

5.18.3 ZIMPOL CCD

The two chosen ZIMPOL CCDs are e2v 44-82 bi, in frame transfer mode (oneCCD 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, theother half (2k×2k) is exposed to light.The exposed part of the CCD is furthermore equipped with an opaque stripemask which alternately covers two rows of the CCD and leaves the next tworows 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 ≈ 0.117 mas/μm 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 imageswill 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≈ 0.233 mas/μm or a pixel scale of 7 mas/pixel.This corresponds to about half a resolution element at 600 nm, which is $\lambda/D \approx 15.5$ 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 5.18.3.There, the boxes with the smallest sizes correspond to the 15 μm sized hardware pixels.The always present on-chip (TBC) 2×2 binning

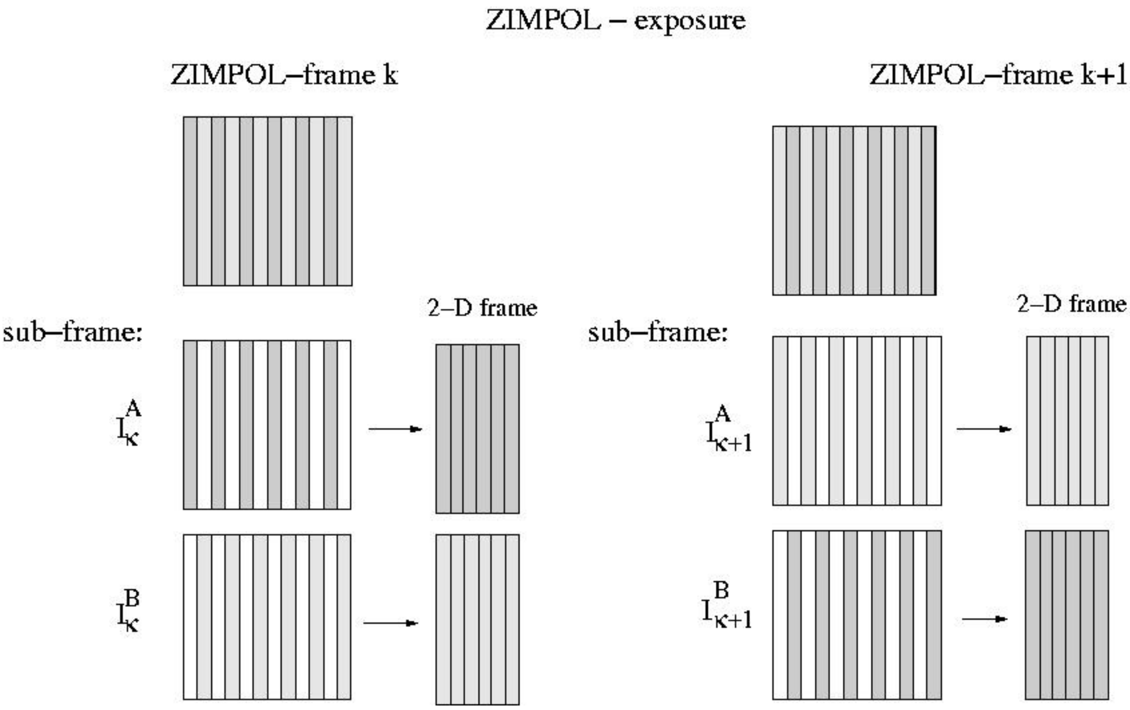

Figure 5.2: Schema of one single ZIMPOL *exposure*for the example of NDIT=1, leading to the output of two consecutive *frames*from each CCD.The left side is the kth ZIMPOL frame recorded at phase one of the double-phase mode, whereasthe right side (frame $k + 1$) has been recorded in the second phase.Each *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 anellipse).Dark grey means more intensity than light grey; white means that thesecolumns contain no scientific data (but only noise).

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 thelargest filled box.A circle with the diameter of diffraction limited resolution is over-plottedto visualize the relations.

5.18.4 Dithering

Dithering is foreseen for all ZIMPOL observations.It will be implemented by keeping the telescope pointed at a fixed positionon 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, yoffsets $(\Delta x_i, \Delta y_i)$ 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 froma menu bar).The numbers are calculated by the INS and the commands are given to themotors 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 offsetsof e.g.2 pixels.For each NDITHER position both HWP2 positions with NDIT frames each canbe taken at the fixed position of FOV (telescope and tilt- and tip/tiltmirrors).The observation sequence is according to the “Observations modes and sequences” (RD4) the following:

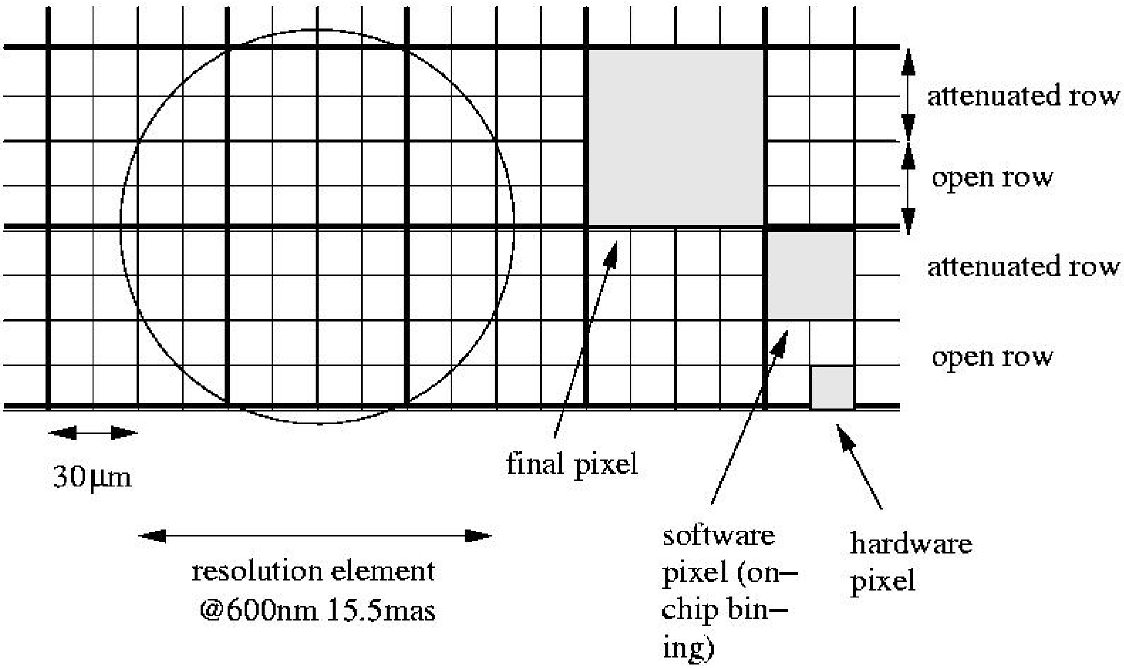

Figure 5.3: Detail of one ZIMPOL CCD with shown resolution element λ/D, the hardware pixel, the 2×2 binned software pixel and the final reduction pixel of 4×4 hardware pixels.

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

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

```
foriDither=0, NDITHER-1 do
    assume dithering position iDither
    setHWP2-offset angle to0
    foriDIT=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
    setHWP2-offset angle to45
    foriDIT=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
```

5.18.5 Two-phase mode

In polarimetric detector modes, two more calibrations must be applied tothe data after dark subtraction and intensity flat-fielding.As described in Section 5.18.2, each ZIMPOL frame $k$consists of two interlaced sub-frames $i^{\mathrm{A}}(k)$, $i^{\mathrm{B}}(k)$that each store one polarization component, e.g. Q_+, Q_-.From one frame to the next, the assignment of polarization components tosub-frames is reversed by switching the phase of the demodulation cyclefrom 0to π, whereas the detector's fixed pattern noise (FPN) remains unchanged.This property is exploited to remove the fixed pattern noise:

$$Q^{(0)} = i_k^{\mathrm{A}} - i_k^{\mathrm{B}} = \frac{1}{2}\left(\left(+Q + FPN^{\mathrm{A}}\right) - \left(-Q + FPN^{\mathrm{B}}\right)\right), \tag{5.11}$$

$$Q^{(\pi)} = i_{k+1}^{\mathrm{A}} - i_{k+1}^{\mathrm{B}} = \frac{1}{2}\left(\left(-Q + FPN^{\mathrm{A}}\right) - \left(+Q + FPN^{\mathrm{B}}\right)\right), \tag{5.12}$$

$$Q = \frac{1}{2}\left(Q^{(0)} - Q^{(\pi)}\right). \tag{5.13}$$

The fixed pattern noise of a given pixel has been assumed to be constanthere.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 polarizationof the science image as low as possible.The ZIMPOL instrument uses a rotatable and tiltable glass plate to compensatethe background polarization in real-time for this purpose.

5.19 Specific ZIMPOL detector calibration

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

5.19.1 Modulation / demodulation efficiency

The modulation / demodulation polarimetric efficiency of theZIMPOL 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 calibrationrather 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/Iscience image is then divided by the resulting Q/Iefficiency frame to remove those effects.

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

$$(Q/I)_{\mathrm{clean}} = \frac{(Q/I)}{(Q/I)_{\mathrm{MDE}}}. \tag{5.14}$$

Chapter 6

Installation Procedure and Troubleshooting

6.1 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 maindirectory

```
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 isyour 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 supportfor the SPHERE pipeline.However, with some small changes it is possible to install the pipelineon 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, youmay 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, pleasedo 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 youinstalled everything (the name you gave when you ran the install_pipelinescript).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
```

6.2 Tips, tricks, and troubleshooting

6.2.1 My recipe run terminats with thousands of error messages...

Don't panic! Try to go through the log file produced by esorex and findthe first ERROR signal.In many cases, this gives a hint what went wrong and what can be triedto correct the behaviour...

6.2.2 I still don't understand the error / it says that the 'actual error waslost'

Now panic! Better still, note that many parts of the pipeline are stilluntested and it's not unlikely you stumbled onto something that has neverbeen tried before.Try to make a small package of data and a commandline that can reproducethe eroror and contact us!

6.2.3 My distortion map or other peak-finding involving recipe doesn't producean 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 particularthe 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
Unit:              1
Comment Field:     Mean value of a master frame [1]
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
Unit:              1
Comment Field:     Median value of a master frame[1]
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
Unit:               1
Comment Field:      RMS of a master frame[1]
Description:        RMS of a master frame

#
# Code define: SPH_COMMON_KEYWORD_QC_RON
#
#    Code references:
#     - sph_ird_gain_run.c
#     - sph_ifs_gain_run.c
#     - sph_gain_and_ron.c
#
Parameter Name:     ESO QC RON
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Unit:               ADU
Comment Field:      Determined read-out noise [ADU]
Description:        Determined read-out noise (ADU)

#
# Code define: SPH_COMMON_KEYWORD_QC_RON_RMS
#
#    Code references:
#     - sph_ird_gain_run.c
#     - sph_ifs_gain_run.c
#     - sph_gain_and_ron.c
#
Parameter Name:     ESO QC RON RMS
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Unit:               ADU
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_ird_gain_run.c
#     - sph_ifs_gain_run.c
#     - sph_gain_and_ron.c
#
Parameter Name:     ESO QC GAIN
Class:              header|qc-log
```

```
Context:            process
Type:               double
Value Format:       %.2f
Unit:               e-/ADU
Comment Field:      Determined gain [e-/ADU]
Description:        Determined gain (e-/ADU)

#
# Code define: SPH_COMMON_KEYWORD_QC_GAIN_RMS
#
#    Code references:
#     - sph_ird_gain_run.c
#     - sph_ifs_gain_run.c
#     - sph_gain_and_ron.c
#
Parameter Name:     ESO QC GAIN RMS
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Unit:               e-/ADU
Comment Field:      Determined RMS of gain [e-/ADU]
Description:        Determined RMS of gain (e-/ADU)

#
# 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
Unit:               ADU
Comment Field:      Mean value of an intensity double image (ZIMPOL)[ADU]
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
Unit:               ADU
Comment Field:      Median value of an intensity double image (ZIMPOL)[ADU]
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
Unit:              ADU
Comment Field:     RMS of an intensity double image (ZIMPOL)[ADU]
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
Unit:              1
Comment Field:     Mean value of a polarization degree double image (ZIMPOL)[1]
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
Unit:              1
Comment Field:     Median value of a polarization degree double image (ZIMPOL)[1]
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
Unit:               1
Comment Field:      RMS of a polarization degree double image (ZIMPOL)[1]
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
Unit:               ADU
Comment Field:      Mean value of an intensity triple image (ZIMPOL)[ADU]
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
Unit:               ADU
Comment Field:      Median value of an intensity triple image (ZIMPOL)[ADU]
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
Unit:               ADU
Comment Field:      RMS of an intensity triple image (ZIMPOL)[ADU]
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
Unit:              1
Comment Field:     Mean value of a Q triple image (ZIMPOL)[1]
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
Unit:              1
Comment Field:     Median value of a Q triple image (ZIMPOL)[1]
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
Unit:              1
Comment Field:     RMS of a Q triple image (ZIMPOL)[1]
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
Unit:              1
```

```
Comment Field:      Mean value of a U triple image (ZIMPOL)[1]
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
Unit:               1
Comment Field:      Median value of a U triple image (ZIMPOL)[1]
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
Unit:               1
Comment Field:      RMS a U triple image (ZIMPOL)[1]
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
Unit:               ADU
Comment Field:      Mean value of a zero-odd quadimage [ADU]
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
Unit:              ADU
Comment Field:     Median value of a zero-odd quadimage [ADU]
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
Unit:              ADU
Comment Field:     RMS of a zero-odd quadimage [ADU]
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
Unit:              ADU
Comment Field:     Mean value of a zero-even quadimage [ADU]
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
Unit:              ADU
Comment Field:     Median value of a zero-even quadimage [ADU]
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
Unit:              ADU
Comment Field:     RMS of a zero-even quadimage [ADU]
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
Unit:              ADU
Comment Field:     Mean value of a pi-odd quadimage [ADU]
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
Unit:              ADU
Comment Field:     Median value of a pi-odd quadimage [ADU]
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
Unit:               ADU
Comment Field:      RMS of a pi-odd quadimage [ADU]
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
Unit:               ADU
Comment Field:      Mean value of a pi-even quadimage [ADU]
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
Unit:               ADU
Comment Field:      Median value of a pi-even quadimage [ADU]
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
Unit:               ADU
Comment Field:      RMS of a pi-evn quadimage [ADU]
Description:        RMS of a pi-even quadimage (ZIMPOL)

#
# Code define: SPH_COMMON_KEYWORD_DARK_MEAN_RON
#
```

```
#    Code references:
#     - sph_ird_master_dark_run.c
#     - sph_ifs_master_dark_run.c
#
Parameter Name:    ESO QC MEAN RON
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              ADU
Comment Field:     mean dark RON estimate [ADU]
Description:       Mean dark RON estimate (ADU)

#
# Code define: SPH_COMMON_KEYWORD_INSBG_MEAN_COUNT
#
#    Code references:
#     - sph_ird_sky_bg_run.c
#     - sph_ird_ins_bg_run.c
#
Parameter Name:    ESO QC INSBG MEAN
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              ADU
Comment Field:     mean instrument background count [ADU]
Description:       Mean instrument background count (ADU)

#
# Code define: SPH_COMMON_KEYWORD_INSBG_RMS_COUNT
#
#    Code references:
#     - sph_ird_ins_bg_run.c
#
Parameter Name:    ESO QC INSBG RMS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              ADU
Comment Field:     mean instrument background RMS [ADU]
Description:       Mean instrument background count (ADU)

#
# Code define: SPH_COMMON_KEYWORD_SKYBG_MEAN_COUNT
#
#
Parameter Name:    ESO QC SKYBG MEAN
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Unit:              ADU
```

```
Comment Field:      mean sky background count [ADU]
Description:        Mean sky background count (ADU)

#
# Code define: SPH_COMMON_KEYWORD_SKYBG_RMS_COUNT
#
#    Code references:
#     - sph_ird_sky_bg_run.c
#
Parameter Name:     ESO QC SKYBG RMS
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Unit:               ADU
Comment Field:      mean sky background RMS [ADU]
Description:        Mean sky background count (ADU)

#
# Code define: SPH_COMMON_KEYWORD_NUMBER_HOTPIXELS
#
#    Code references:
#     - sph_ird_sky_bg_run.c
#     - sph_ird_ins_bg_run.c
#     - sph_ird_master_dark_run.c
#     - sph_ifs_master_dark_run.c
#
Parameter Name:     ESO QC NUM HOTPIXELS
Class:              header|qc-log
Context:            process
Type:               int
Value Format:       %d
Unit:               1
Comment Field:      No. of identified hot pixels [1]
Description:        Number of identified hot pixels

#
# Code define: SPH_COMMON_KEYWORD_FLAT_FPN
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#     - sph_ird_tff_run.c
#     - sph_ifs_instrument_flat_run.c
#     - sph_ifs_master_detector_flat_run.c
#
Parameter Name:     ESO QC FPN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Unit:               1
Comment Field:      Fixed pattern noise [1]
Description:        Fixed pattern noise
```

```
#
# Code define: SPH_COMMON_KEYWORD_FLAT_RMS
#
#    Code references:
#      - sph_ird_instrument_flat_run.c
#      - sph_ird_tff_run.c
#
Parameter Name:    ESO QC RMS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              1
Comment Field:     Root mean squared [1]
Description:       Root mean squared

#
# Code define: SPH_COMMON_KEYWORD_FLAT_NONLIN_FACTOR
#
#    Code references:
#      - sph_ird_instrument_flat_run.c
#      - sph_ird_tff_run.c
#      - sph_ifs_instrument_flat_run.c
#      - sph_ifs_master_detector_flat_run.c
#
Parameter Name:    ESO QC FLAT NONLINEARITY
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              1
Comment Field:     nonlinearity coefficient [1]
Description:       nonlinearity coefficient

#
# Code define: SPH_COMMON_KEYWORD_FLAT_MEAN_COUNT
#
#    Code references:
#      - sph_ird_instrument_flat_run.c
#      - sph_ird_tff_run.c
#      - sph_ifs_instrument_flat_run.c
#      - sph_ifs_master_detector_flat_run.c
#
Parameter Name:    ESO QC FLAT MEAN
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              ADU
Comment Field:     Mean counts in flat field [ADU]
Description:       Mean counts in flat field

#
# Code define: SPH_COMMON_KEYWORD_NUMBER_BADPIXELS
```

```
#
#    Code references:
#      - sph_ird_instrument_flat_run.c
#      - sph_ird_tff_run.c
#      - sph_zpl_intensity_flat_run.c
#      - sph_ifs_instrument_flat_run.c
#      - sph_ifs_master_detector_flat_run.c
#      - sph_gain_and_ron.c
#
Parameter Name:    ESO QC NUM BADPIXELS
Class:             header|qc-log
Context:           process
Type:              int
Value Format:      %d
Unit:              1
Comment Field:     No. of identified bad pixels [1]
Description:       number of identified bad pixels

#
# Code define: SPH_COMMON_KEYWORD_FLAT_LAMP_FLUX
#
#    Code references:
#      - sph_ifs_instrument_flat_run.c
#      - sph_ifs_master_detector_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
Unit:              ADU/s
Comment Field:     Lamp flux in single frame [ADU/s]
Description:       Lamp flux (counts/s in single frame)

#
# Code define: SPH_COMMON_KEYWORD_FLAT_LAMP_COUNTS
#
#    Code references:
#      - sph_ifs_instrument_flat_run.c
#
Parameter Name:    ESO QC LAMP ADU AVG
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              ADU
Comment Field:     Lamp counts in single frame [ADU]
Description:       Lamp ADU (counts in single frame)

#
# Code define: SPH_COMMON_KEYWORD_FLAT_LAMP_FLUX_STDEV
#
#    Code references:
```

```
#      - sph_ifs_instrument_flat_run.c
#      - sph_ifs_master_detector_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
Unit:               ADU/s
Comment Field:      Lamp flux stdev in single frame [ADU/s]
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
Unit:               1
Comment Field:      median resolving power [1]
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
Unit:               micron/px
Comment Field:      median dispersion [micron/px]
Description:        median value of dispersion

#
# 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
```

```
Unit:               micron
Comment Field:      median value of longest wavelength [micron]
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
Unit:               1
Comment Field:      no. of wavelengths out of 1 dispersion [1]
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
Unit:               1
Comment Field:      no. of bad spectra [1]
Description:        no. of bad spectra

#
# Code define: SPH_COMMON_KEYWORD_DISTMAP_NREMOVED
#
#    Code references:
#     - sph_distortion_model.c
#
Parameter Name:     ESO QC DISTMAP NREMOVED
Class:              header|qc-log
Context:            process
Type:               int
Value Format:       %d
Unit:               1
Comment Field:      no. of points removed from distortion map [1]
Description:        no. of points removed from distortion map

#
# Code define: SPH_COMMON_KEYWORD_DISTMAP_PIXSCALE
#
#    Code references:
#     - sph_distortion_model.c
```

```
#
Parameter Name:    ESO QC DISTMAP PIXSCALE
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              1
Comment Field:     relative distortion map pixel scale [1]
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
Unit:              1
Comment Field:     number of bad pixels[1]
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
Unit:              1
Comment Field:     number of bad pixels [1]
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
Unit:              1
Comment Field:     number of bad pixels [1]
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
Unit:               1
Comment Field:      number of bad pixels [1]
Description:        number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_QUAD_IMAGE_ZERO_ODD
#
#    Code references:
#     - sph_zpl_intensity_flat_run.c
#     - sph_zpl_master_bias_run.c
#     - sph_zpl_master_dark_run.c
#     - sph_quad_image.c
#
Parameter Name:     ESO QC QUAD IMAGE ZERO ODD NUMBER BADPIXELS
Class:              header|qc-log
Context:            process
Type:               long
Value Format:       %.2f
Unit:               1
Comment Field:      number of bad pixels [1]
Description:        number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_QUAD_IMAGE_ZERO_EVEN
#
#    Code references:
#     - sph_zpl_intensity_flat_run.c
#     - sph_zpl_master_bias_run.c
#     - sph_zpl_master_dark_run.c
#     - sph_quad_image.c
#
Parameter Name:     ESO QC QUAD IMAGE ZERO EVEN NUMBER BADPIXELS
Class:              header|qc-log
Context:            process
Type:               long
Value Format:       %.2f
Unit:               1
Comment Field:      number of bad pixels [1]
Description:        number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_QUAD_IMAGE_PI_ODD
```

```
#
#    Code references:
#     - sph_zpl_intensity_flat_run.c
#     - sph_zpl_master_bias_run.c
#     - sph_zpl_master_dark_run.c
#     - sph_quad_image.c
#
Parameter Name:    ESO QC QUAD IMAGE PI ODD NUMBER BADPIXELS
Class:             header|qc-log
Context:           process
Type:              long
Value Format:      %.2f
Unit:              1
Comment Field:     number of bad pixels [1]
Description:       number of bad pixels

#
# Code define: SPH_COMMON_KEYWORD_QC_NUMBER_BADPIXELS_QUAD_IMAGE_PI_EVEN
#
#    Code references:
#     - sph_zpl_intensity_flat_run.c
#     - sph_zpl_master_bias_run.c
#     - sph_zpl_master_dark_run.c
#     - sph_quad_image.c
#
Parameter Name:    ESO QC QUAD IMAGE PI EVEN NUMBER BADPIXELS
Class:             header|qc-log
Context:           process
Type:              long
Value Format:      %.2f
Unit:              1
Comment Field:     number of bad pixels [1]
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
Unit:              1
Comment Field:     number of bad pixels [1]
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
Unit:              1
Comment Field:     number of bad pixels [1]
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:              long
Value Format:      %.2f
Unit:              1
Comment Field:     number of bad pixels [1]
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:              long
Value Format:      %.2f
Unit:              1
Comment Field:     number of bad pixels [1]
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
Unit:              1
```

```
Comment Field:      number of rings used by LOCI [1]
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
Unit:               1
Comment Field:      number of sub-sections used by LOCI [1]
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
Unit:               1
Comment Field:      number of bad sub-sections [1]
Description:        number of bad sub-sections

#
# Code define: SPH_STREHL_QC_STREHL
#
#
Parameter Name:     ESO QC STREHL
Class:              header|qc-log
Context:            process
Type:               string
Value Format:       %30s
Unit:               1
Comment Field:      Strehl ratio measured on brightest source in field
Description:        Strehl ratio measured on brightest source in field

#
# Code define: SPH_STREHL_QC_STREHL_ERR
#
#
Parameter Name:     ESO QC STREHL ERROR
Class:              header|qc-log
Context:            process
Type:               string
```

```
Value Format:      %30s
Unit:              1
Comment Field:     Error of Strehl ratio measured on brightest source in field
Description:       Error of Strehl ratio measured on brightest source in field

#
# Code define: SPH_STREHL_QC_STREHL_POSX
#
#
Parameter Name:    ESO QC STREHL POSX
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Unit:              1
Comment Field:     X coordinate of position of source used for Strehl measurement
Description:       X coordinate of position of source used for Strehl measurement

#
# Code define: SPH_STREHL_QC_STREHL_POSY
#
#
Parameter Name:    ESO QC STREHL POSY
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Unit:              1
Comment Field:     Y coordinate of position of source used for Strehl measurement
Description:       Y coordinate of position of source used for Strehl measurement

#
# Code define: SPH_STREHL_QC_STREHL_SIGMA
#
#
Parameter Name:    ESO QC STREHL SIGMA
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Unit:              1
Comment Field:     Sigma of Strehl measurement (?)
Description:       Sigma of Strehl measurement

#
# Code define: SPH_STREHL_QC_STREHL_FLUX
#
#
Parameter Name:    ESO QC STREHL FLUX
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Unit:              ADU
```

```
Comment Field:     Flux measured in aperture for Strehl measurement [ADU]
Description:       Flux measured in aperture for Strehl measurement

#
# Code define: SPH_STREHL_QC_STREHL_PEAK
#
#
Parameter Name:    ESO QC STREHL PEAK
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Unit:              ADU
Comment Field:     Peak intensity in aperture used for Strehl measurement [ADU]
Description:       Peak intensity in aperture used for Strehl measurement

#
# Code define: SPH_STREHL_QC_STREHL_BKG
#
#
Parameter Name:    ESO QC STREHL BACKGROUND
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Unit:              ADU
Comment Field:     Background flux per pixel in aperture used for Strehl measurement [ADU]
Description:       Background flux per pixel in aperture used for Strehl measurement

#
# Code define: SPH_STREHL_QC_STREHL_BKGNOISE
#
#
Parameter Name:    ESO QC STREHL BACKGROUND NOISE
Class:             header|qc-log
Context:           process
Type:              string
Value Format:      %30s
Unit:              ADU
Comment Field:     Noise of background flux per pixel in aperture used for Strehl measurement [AD
Description:       Noise of background flux per pixel in aperture used for Strehl measurement
```

A.2 IRDIS

```
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_NGOODLINES
#
#    Code references:
#     - sph_ird_wave_calib_run.c
#
Parameter Name:    ESO QC WAVECAL NGOODCOLUMNS
```

```
Class:              header|qc-log
Context:            process
Type:               int
Value Format:       %d
Unit:               1
Comment Field:      Number of good spectral regions / Columns [1]
Description:        Number of good columns

#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_NBADLINES
#
#    Code references:
#     - sph_ird_wave_calib_run.c
#
Parameter Name:     ESO QC WAVECAL NBADCOLUMNS
Class:              header|qc-log
Context:            process
Type:               int
Value Format:       %d
Unit:               1
Comment Field:      Number of bad spectral regions / columns [1]
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_NNOFITLINES
#
#    Code references:
#     - sph_ird_wave_calib_run.c
#
Parameter Name:     ESO QC WAVECAL NNOFITCOLUMNS
Class:              header|qc-log
Context:            process
Type:               int
Value Format:       %d
Unit:               1
Comment Field:      Number of spectral regions/columns wothout fit [1]
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_Y0_MEAN
#
#    Code references:
#     - sph_ird_wave_calib_run.c
#
Parameter Name:     ESO QC WAVECAL Y0 MEAN
Class:              header|qc-log
Context:            process
Type:               double
Value Format:       %.2f
Unit:               micron
Comment Field:      Mean minimum wavelength [micron]
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_Y0_RMS
#
#    Code references:
```

```
#      - sph_ird_wave_calib_run.c
#
Parameter Name:    ESO QC WAVECAL Y0 RMS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              micron
Comment Field:     RMS of minimum wavelength [micron]
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_C1_MEAN
#
#     Code references:
#      - sph_ird_wave_calib_run.c
#
Parameter Name:    ESO QC WAVECAL C1 MEAN
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              micron/pix
Comment Field:     Mean slope of wavelength solution [micron/pix]
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_C1_RMS
#
#     Code references:
#      - sph_ird_wave_calib_run.c
#
Parameter Name:    ESO QC WAVECAL C1 RMS
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              micron/pix
Comment Field:     RMS of mean slope of wavelength solution [micron/pix]
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_Y0
#
#     Code references:
#      - sph_ird_wave_calib_run.c
#
Parameter Name:    ESO QC WAVECAL Y0 COL
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              micron
Comment Field:     Minimum wavelength for spectral region/column [micron]
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_C1
```

```
#
#    Code references:
#     - sph_ird_wave_calib_run.c
#
Parameter Name:    ESO QC WAVECAL C1 COL
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              micron/pix
Comment Field:     Slope of wavelength solution for spectral region/column [micron/pix]
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_CHI
#
#    Code references:
#     - sph_ird_wave_calib_run.c
#
Parameter Name:    ESO QC WAVECAL CHI2 COL
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              micron2
Comment Field:     CHI2 of wavelength solution for spectral region/column [micron2]
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_QC_LAM
#
#    Code references:
#     - sph_ird_wave_calib_run.c
#
Parameter Name:    ESO QC WAVECAL LAM LINE
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              micron
Comment Field:     Wavelength of line [micron]
Description:
#
# Code define: SPH_IRD_KEYWORD_WAVECALIB_QC_LAMRMS
#
#    Code references:
#     - sph_ird_wave_calib_run.c
#
Parameter Name:    ESO QC WAVECAL LAM RMS LINE
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              micron
Comment Field:     RMS of line wavelength [micron]
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
Unit:               ADU
Comment Field:      Total star flux in left coro image [ADU]
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
Unit:               ADU
Comment Field:      Total star flux in right coro image [ADU]
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
Unit:               ADU
Comment Field:      Total star flux in left non-coro image [ADU]
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
Unit:               ADU
```

```
Comment Field:      Total star flux in right non-coro image [ADU]
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
Unit:               1
Comment Field:      Number of points found in observed pattern [1]
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
Unit:               1
Comment Field:      Number of points expected in input pattern [1]
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
Unit:               px
Comment Field:      X position of optical axis [px]
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
Unit:              px
Comment Field:     Y position of optical axis [px]
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
Unit:              perc. of mean
Comment Field:     FPN of left frame [perc. of mean]
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
Unit:              1
Comment Field:     RMS of left frame [1]
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
Unit:              1
Comment Field:     Non-linearity factor of left frame [1]
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
Unit:               ADU
Comment Field:      Mean value of left frame [ADU]
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
Unit:               perc. of mean
Comment Field:      FPN of right frame [perc. of mean]
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
Unit:               1
Comment Field:      RMS of right frame [1]
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
Unit:               1
Comment Field:      Non-linearity factor of right frame [1]
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
Unit:              ADU
Comment Field:     Mean value of right frame [ADU]
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
Unit:              ADU/s
Comment Field:     Average lamp flux in left frame [ADU/s]
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
Unit:              ADU/s
Comment Field:     Average lamp flux in right frame [ADU/s]
Description:
#
# Code define: SPH_IRD_KEYWORD_FLAT_LAMP_COUNTS_LEFT
#
#    Code references:
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:    ESO QC LAMP ADU AVG LEFT
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              ADU/s
Comment Field:     Average lamp flux in right frame [ADU/s]
Description:
#
# Code define: SPH_IRD_KEYWORD_FLAT_LAMP_COUNTS_RIGHT
#
#    Code references:
```

```
#     - sph_ird_instrument_flat_run.c
#
Parameter Name:    ESO QC LAMP ADU AVG RIGHT
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              ADU
Comment Field:     Average total lamp counts in right frame [ADU]
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
Unit:              ADU/s
Comment Field:     SDEV of avg. lamp flux in left frame [ADU/s]
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
Unit:              ADU/s
Comment Field:     SDEV of avg. lamp flux in right frame [ADU/s]
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
```

```
Unit:               1
Comment Field:      Number of spectra found [1]
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
Unit:               ADU
Comment Field:      Threshold used for spectra detection [ADU]
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
Unit:               1
Comment Field:      Scale of specpos model as measured [1]
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
Unit:               ADU
Comment Field:      background mean value [ADU]
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
Unit:              ADU
Comment Field:     Background RMS [ADU]
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
Unit:              ADU
Comment Field:     Background median value [ADU]
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
Unit:              1
Comment Field:     Mean value of preamp correlation [1]
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
Unit:              1
Comment Field:     Median value of preamp correlation [1]
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
Unit:              1
Comment Field:     RMS of preamp correlation [1]
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
Unit:              1
Comment Field:     Number of points observed for distortion map [1]
Description:       Number of points observed for distortion map

#
# Code define: SPH_IFS_KEYWORD_DISTMAP_POLFIT_CHIX
#
#    Code references:
#     - sph_ifs_spectra_positions_run.c
#
Parameter Name:    ESO QC DISTMAP POLFIT CHIX
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              1
Comment Field:     Chi squared of the polynomial distortion fit [1]
Description:       Red. chi squared of the polynomial distortion fit

#
# Code define: SPH_IFS_KEYWORD_DISTMAP_POLFIT_CHIY
#
#    Code references:
#     - sph_ifs_spectra_positions_run.c
#
Parameter Name:    ESO QC DISTMAP POLFIT CHIY
Class:             header|qc-log
Context:           process
Type:              double
Value Format:      %.2f
Unit:              1
Comment Field:     Chi squared of the polynomial distortion fit [1]
Description:       Red. chi squared of the polynomial distortion fit
```

A.4 ZIMPOL

There are currently no specific ZIMPOL QC keywords.See common.