SPIRou Data Reduction Software

Developer Guide

0.0.1

For DRS SPIRou0.0.1

N. Cook, F. Bouchy, E. Artigau, I. Boisse, M. Hobson, C. Moutou

2017-12-08

Abstract

This is the guide to coding the DRS (including installation, running, rules and stardisation approaches). This document is not intended for the general used of the DRS, instead it is intended for those who wish to develop the software further and understand the changes between this version and previous versions.

Contents

1 Introduction **1**

2 Installation **3**
2.1 Introduction . . . 3
2.2 Download . . . 3
2.3 Prerequisites . . . 4
2.3.1 Anaconda python distribution . . . 4
2.3.2 Separate python installation . . . 4
2.4 Installation Linux and macOS . . . 5
2.4.1 Extraction . . . 5
2.4.2 Modify environmental settings . . . 5
2.4.3 Make recipes executable . . . 5
2.5 Installation Windows . . . 6
2.5.1 How to modify environmental settings in windows . . . 6
2.6 Setting up the DRS . . . 7
2.7 Validating Installation on Linux and macOS . . . 8
2.8 Validating Installation on Windows . . . 9

3 Data Architecture **10**
3.1 Installed file structure . . . 10
3.2 The Installation root directory . . . 11
3.2.1 The bin directory . . . 11
3.2.2 The SPIROU module directory . . . 11
3.3 The data root directory . . . 12
3.3.1 The raw and reduced data directories . . . 12
3.4 The calibration database directory . . . 13

4 Using the DRS **14**
4.1 Running the DRS recipes directly . . . 14
4.2 Running the DRS recipes from a python script . . . 14
4.3 Working example of the code for SPIRou . . . 14
4.3.1 Overview . . . 14
4.3.2 Run through from command line/python shell (Linux and macOS) . . . 15
4.3.3 Run through python script . . . 18

5 Summary of changes (AT-4) **19**
5.1 General . . . 19
5.2 The cal_DARK_spirou recipe . . . 20
5.3 The cal_loc_RAW_spirou recipe . . . 20
5.4 The cal_SLIT_spirou recipe . . . 21
5.5 The cal_FF_RAW_spirou recipe . . . 22
5.6 The cal_extract_RAW_spirou recipes . . . 22
5.7 The cal_DRIFT_RAW_spirou recipe . . . 24
5.8 The cal_BADPIX_spirou recipe . . . 25
5.9 The cal_DRIFT-E2DS_spirou recipe . . . 25
5.10 The cal_DRIFT-PEAK_E2DS_spirou recipe . . . 26

6 Current to do list 27
6.1 General . . . 27
6.2 Documentation . . . 27
6.3 The cal_DARK_spirou recipe . . . 28
6.4 The cal_loc_RAW_spirou recipe . . . 28
6.5 The cal_SLIT_spirou recipe . . . 28
6.6 The cal_FF_RAW_spirou recipe . . . 28
6.7 The cal_extract_RAW_spirou recipes . . . 29
6.8 The cal_DRIFT_RAW_spirou recipe . . . 29
6.9 The cal_BADPIX_spirou recipe . . . 29
6.10 The cal_DRIFT_E2DS_spirou recipe . . . 29
6.11 The cal_DRIFT-PEAK_E2DS_spirou recipe . . . 29
6.12 The cal_HC_E2DS_spirou recipe . . . 30
6.13 The cal_WAVE_E2DS_spirou recipe . . . 30
6.14 The cal_CCF_E2DS_spirou recipe . . . 30
6.15 The pol_spirou recipe . . . 30

7 Coding style and standardization 31
7.1 PEP 8 - A style guide for python code . . . 31
7.2 DRS specific style and standardization . . . 32
7.2.1 Functions from sub-modules . . . 32
7.2.2 The logger (WLOG) . . . 33
7.2.3 The Parameter Dictionary Object . . . 34
7.2.4 Configuration Error and Exception . . . 36

8 Writing the documentation 37
8.1 Documentation Architecture . . . 37
8.2 Required LaTeXpackages . . . 37
8.3 Developer documentation content . . . 38
8.4 Custom Commands . . . 39
8.5 Constants . . . 44
8.6 Code formatting . . . 44

9 Required input header keywords 48
9.1 Standard FITS Keywords . . . 48
9.2 FITS keywords related to the detector . . . 48
9.3 FITS keywords related to the target . . . 48
9.4 FITS keywords related to the telescope . . . 49
9.5 FITS keywords related to the instrument . . . 49

10 Variables 50
10.1 Variable file locations . . . 50
10.1.1 User modifiable variables . . . 50
10.1.2 Private variables . . . 50
10.2 Global variables . . . 51
10.3 Directory variables . . . 54
10.4 Image variables . . . 56
10.5 Fiber variables . . . 58
10.6 Dark calibration variables . . . 62
10.7 Localization calibration variables . . . 64
10.8 Slit calibration variables . . . 72
10.9 Flat fielding calibration variables . . . 74
10.10 Extraction calibration variables . . . 77

10.11 Drift calibration variables . . . 80
10.12 Drift-Peak calibration variables . . . 84
10.13 Bad pixel calibration variables . . . 89
10.14 Quality control variables . . . 91
10.15 Calibration database variables . . . 95
10.16 Startup variables . . . 96
10.17 Formatting variables . . . 100
10.18 FITS rec variables . . . 101
10.19 Logging and printing variables . . . 102

11 Output header keywords **106**
11.1 Global keywords . . . 107
11.2 Dark calibration keywords . . . 109
11.3 Localization calibration keywords . . . 112
11.4 Slit calibration keywords . . . 117
11.5 Flat fielding calibration keywords . . . 117
11.6 Extraction calibration keywords . . . 118
11.7 Bad pixel calibration keywords . . . 118

12 The Recipes **120**
12.1 The cal_DARK recipe . . . 120
12.2 The cal_loc recipe . . . 120
12.3 The cal_SLIT recipe . . . 120
12.4 The cal_FF recipe . . . 120
12.5 The cal_extract recipes . . . 120
12.6 The cal_DRIFT recipes . . . 120
12.7 The cal_BADPIX recipe . . . 120
12.8 The cal_HC recipe . . . 120
12.9 The cal_WAVE recipe . . . 120
12.10 The cal_CCF recipe . . . 120
12.11 The pol_spirou recipe . . . 120

13 The DRS Module **121**
13.1 The spirouBACK module . . . 121
13.2 The spirouCDB module . . . 121
13.3 The spirouCore module . . . 121
13.4 The spirouEXTOR module . . . 121
13.5 The spirouFLAT module . . . 121
13.6 The spirouImage module . . . 121
13.7 The spirouLOCOR module . . . 121
13.8 The spirouRV module . . . 121
13.9 The spirouStartup module . . . 121
13.10 The spirouUnitTests module . . . 121

Chapter 1

Introduction

This documentation will cover the installation, data architecture, the changes between the previous versions and this version, using the DRS (with a working example), descriptions of the variables and keywords for input and output FITS rec headers , and the recipes and module code .

Variables are defined in detail in section 10 and will be defined throughout via the following syntax: {VARIABLE}. When refered to, one should take it as using the value set in section 10 by default or in the file described in the variables description 'Defined in' section.

Certain sections will be written in code blocks, these imply text that is written into a text editor, the command shell console, or a python terminal/script. Below explains how one can distinguish these in this document.

The following denotes a line of text (or lines of text) that are to be edited in a text editor.

text

```
# A variable name that can be changes to a specific value
VARIABLE_NAME = "Variable Value"
```

These can also be shell scripts in a certain language:

bash

```
#!/usr/bin/bash
# Find out which console you are using
echo $0
# Set environment Hello
export Hello="Hello"
```

tcsh/csh

```
#!/usr/bin/tcsh
# Find out which console you are using
echo $0
# Set environment Hello
setenv Hello "Hello"
```

The following denotes a command to run in the command shell console

CMD input

```
>> cd ~/Downloads
```

The following denotes a command line print out

Command line output

```
This is a print out in the command line
produced by using the echo command
```

The following denotes a python terminal or python script

Python/Ipython

```
import numpy as np
print("Hello world")
print("{0} seconds".format(np.sqrt(25)))
```

The following denotes LaTeXcode (in raw form and then compiled form) - this is used in Section 8.

LaTeX

```
This is my \LaTeX code.
```

This is my LaTeXcode.

Chapter 2

Installation

2.1 Introduction

Once finalized the installation should just be a download, run setup.py and configure the DRS directories, however, during development the following stages are required.

> **Note:** Currently the download repository on git-hub is private and requires a git-hub account, and the user to be added to the list of collaborators. To be added to the collaborators please email neil.james.cook@gmail.com with your git-hub user name.

2.2 Download

Get the latest version of the DRS (for SPIRouversion 0.0.1). Use any of the following ways:

- manually download from here: https://github.com/njcuk9999/spirou_py3
- use Git:

```
CMD input
>> git checkout https://github.com/njcuk9999/spirou_py3.git
```

- use SVN:

```
CMD input
>> svn checkout  https://github.com/njcuk9999/spirou_py3.git
```

- use ssh:

```
CMD input
>> scp -r git@github.com:njcuk9999/spirou_py3.git
```

2.3 Prerequisites

It is recommended to install the latest version of Anaconda python distribution, available for Windows, macOS and Linux (here: https://www.anaconda.com/download/). However one can run the DRS on a native python installation.

We recommend python 3 over python 2 for long term continued support (however the latest version of the DRS supports the newest versions of python 2.7).

Note: Before installing the DRS you must have one of the following:

- Latest version of Anaconda (for python 2 or python 3) — RECOMMENDED
- An Up-to-date version of python (python 2 or python 3)

2.3.1 Anaconda python distribution

A valid version of the Anaconda python distribution (for python2 or python 3) Currently tested version of python are:

- Python 2.7.13 and Anaconda 4.4.0
- Python 3.6.3 and Anaconda 5.0.1 — RECOMMENDED

2.3.2 Separate python installation

An up-to-date version of python (either python 2 or python 3) and the following python modules (with version of python they were tested with).

- Python 3.6
  - ASTROPY (tested with version 2.0.2)
  - MATPLOTLIB (tested with version 2.1.0)
  - NUMPY (tested with version 1.13.3)
  - and the following built-in modules (comes with python): DATETIME, FILECMP, GLOB, OS, PKG_RESOURCES, SHUTIL, SYS, TIME, WARNINGS
- Python 2.7
  - astropy (tested with version 1.3.2)
  - matplotlib (tested with version 2.0.2)
  - numpy (tested with version 1.12.1)
  - and the following built-in modules (comes with python): __FUTURE__, COLLECTIONS, DATETIME, FILECMP, GLOB, OS, PKG_RESOURCES, SHUTIL, SYS, TIME, WARNINGS

2.4 Installation Linux and macOS

Currently the DRS has to be installed manually. This involves the following steps:

1. Extraction (Section 2.4.1)
2. Modify environmental settings (Section 2.4.2)
3. Make recipes executable (Section 2.4.3)

2.4.1 Extraction

The first step is to extract the DRS into a folder (the {INSTALL_DIR}).
Do this by using the following commands:

CMD input

```
>> cd {INSTALL_DIR}
>> unzip DRS.zip
```

2.4.2 Modify environmental settings

The next step is to modify your PATH and PYTHONPATH environmental variables (to include the {INSTALL_DIR}. This depends which shell you are using (type '`echo $0`' to find out which).

- In bash open the '.bashrc' text file in your home (~) directory (or create it if it doesn't exist)

bash

```
export PATH={INSTALL_DIR}/bin/:$PATH

export PYTHONPATH={INSTALL_DIR}:{INSTALL_DIR}/bin/:$PYTHONPATH
```

- In csh /tcsh open the '.cshrc' or '.tcshrc' text file in your home (~) directory (or create it if it doesn't exist)

tcsh/csh

```
setenv PATH {INSTALL_DIR}/bin/:${PATH}

setenv PYTHONPATH {INSTALL_DIR}:{INSTALL_DIR}/bin/:${PYTHONPATH}
```

2.4.3 Make recipes executable

To run the recipes from the command line (without starting python) one must make them executable.
Do this by using the following command:

CMD input

```
>> chmod +x {INSTALL_DIR}/bin/*.py
```

2.5 Installation Windows

This is very similar currently to the Linux/macOS installation (in the future a '.exe' file will be given).

1. Extract to {INSTALL_DIR}with your favourite unzipping softwear.
2. Add {INSTALL_DIR}to your PYTHONPATH (Section 2.5.1)

2.5.1 How to modify environmental settings in windows

This process is a little more convoluted than on Linux or macOS system.

1. Go to 'My computer > Properties > Advanced System Settings > Enviromental Variables'.
2. if under system variable 'PythonPath' exists click edit and add '{INSTALL_DIR};' to the end. i.e.

```text
C:\Python27;{INSTALL_DIR};
```

3. if under system variables 'PythonPath' does not exist create a new variable called 'PythonPath' and add:

```text
%PYTHONPATH%;{INSTALL_DIR};{INSTALL_DIR}\bin\;
```

For problems/troubleshooting see here: https://stackoverflow.com/questions/3701646.

2.6 Setting up the DRS

Before running the DRS one must set the data paths.

The '../config/config.txt' file is located in the {INSTALL_DIR}in the config folder.
i.e. at {INSTALL_DIR}/config/../config/config.txt

The following keywords **must** be changed (and must be a valid path):

{TDATA}	=	/drs/data/	/	Define the DATA directory
{DRS_ROOT}	=	/drs/INTROOT/	/	Define the installation directory ({INSTALL_DIR})
{DRS_DATA_RAW}	=	/drs/data/raw	/	Define the folder with the raw data files in
{DRS_DATA_REDUC}	=	/drs/data/reduced	/	Define the directory that the reduced data should be saved to/read from
{DRS_CALIB_DB}	=	/drs/data/calibDB	/	Define the directory that the calibration files should be saved to/read from
{DRS_DATA_MSG}	=	/drs/data/msg	/	Define the directory that the log messages are stored in
{DRS_DATA_WORKING}	=	/drs/data/tmp/	/	Define the working directory

The directories here are for linux and macOS systems another example would be '/home/user/INTROOT' for the {INSTALL_DIR} directory.
On Windows machines this would be equivalent to 'C:\Users\<username>\INTROOT' in Windows Vista, 7, 8 and 10 or 'C:\Documents and Settings\<username>\INTROOT' on early versions of Windows.

The following keywords can be changed:

{DRS_PLOT}	=	1	/	Whether to show plots
{PRINT_LEVEL}	=	"all"	/	Level at which to print
{LOG_LEVEL}	=	"all"	/	Level at which to log in log file

For the '{PRINT_LEVEL} and {LOG_LEVEL} keywords the values are set as follows:

- "all" – prints all events
- "info" – prints info, warning and error events
- "warning" – prints warning and error events
- "error" – print only error events

2.7 Validating Installation on Linux and macOS

> **Note:** One must install the DRS (Section 2.4) AND set up the DRS (Section 2.6) before validation will be successful.

There are four ways to run the DRS in Linux and macOS (thus four ways to verify installation was correct).

- To validate running from command line type:

CMD input

```
>> cal_validate_spirou
```

- To validate running from python/ipython from the command line type:

CMD input

```
>> python cal_validate_spirou
>> ipython cal_validate_spirou
```

- To validate running from ipython, open ipython and type:

Python/Ipython

```
run cal_validate_spirou
```

- To validate running from import from python/ipython, open python/ipython and type:

Python/Ipython

```
import cal_validate_spirou
cal_validate_spirou.main()
```

If validation is successful the following should appear:

Command line output

```
HH:MM:SS.S -   || ****************************************
HH:MM:SS.S -   || * SPIROU @(#) Geneva Observatory (0.0.1)
HH:MM:SS.S -   || ****************************************
HH:MM:SS.S -   ||(dir_data_raw)      DRS_DATA_RAW=/scratch/Projects/spirou_py3/data/raw
HH:MM:SS.S -   ||(dir_data_reduc)    DRS_DATA_REDUC=/scratch/Projects/spirou_py3/data/reduced
HH:MM:SS.S -   ||(dir_calib_db)      DRS_CALIB_DB=/scratch/Projects/spirou_py3/data/calibDB
HH:MM:SS.S -   ||(dir_data_msg)      DRS_DATA_MSG=/scratch/Projects/spirou_py3/data/msg
HH:MM:SS.S -   ||(print_level)       PRINT_LEVEL=all         %(error/warning/info/all)
HH:MM:SS.S -   ||(log_level)         LOG_LEVEL=all         %(error/warning/info/all)
HH:MM:SS.S -   ||(plot_graph)        DRS_PLOT=1            %(def/undef/trigger)
HH:MM:SS.S -   ||(used_date)         DRS_USED_DATE=undefined
HH:MM:SS.S -   ||(working_dir)       DRS_DATA_WORKING=/scratch/Projects/spirou_py3/data/tmp/
HH:MM:SS.S -   ||                    DRS_INTERACTIVE is not set, running on-line mode
HH:MM:SS.S -   ||
HH:MM:SS.S -   ||Validation successful. DRS installed corrected.
```

2.8 Validating Installation on Windows

> **Note:** One must install the DRS (Section 2.5) AND set up the DRS (Section 2.6) before validation will be successful.

In windows there are currently 3 ways to run the RS (running in python/ipython).

- To validate running from python/ipython from the command line type:

CMD input

```
>> python cal_validate_spirou
>> ipython cal_validate_spirou
```

- To validate running from ipython, open ipython and type:

Python/Ipython

```
run cal_validate_spirou
```

- To validate running from import from python/ipython, open python/ipython and type:

Python/Ipython

```
import cal_validate_spirou
cal_validate_spirou.main()
```

If validation is successful the following should appear:

Command line output

```
HH:MM:SS.S -   || ****************************************
HH:MM:SS.S -   || * SPIROU @(#) Geneva Observatory (0.0.1)
HH:MM:SS.S -   || ****************************************
HH:MM:SS.S -   ||(dir_data_raw)      DRS_DATA_RAW=/scratch/Projects/spirou_py3/data/raw
HH:MM:SS.S -   ||(dir_data_reduc)    DRS_DATA_REDUC=/scratch/Projects/spirou_py3/data/reduced
HH:MM:SS.S -   ||(dir_calib_db)      DRS_CALIB_DB=/scratch/Projects/spirou_py3/data/calibDB
HH:MM:SS.S -   ||(dir_data_msg)      DRS_DATA_MSG=/scratch/Projects/spirou_py3/data/msg
HH:MM:SS.S -   ||(print_level)       PRINT_LEVEL=all         %(error/warning/info/all)
HH:MM:SS.S -   ||(log_level)         LOG_LEVEL=all         %(error/warning/info/all)
HH:MM:SS.S -   ||(plot_graph)        DRS_PLOT=1            %(def/undef/trigger)
HH:MM:SS.S -   ||(used_date)         DRS_USED_DATE=undefined
HH:MM:SS.S -   ||(working_dir)       DRS_DATA_WORKING=/scratch/Projects/spirou_py3/data/tmp/
HH:MM:SS.S -   ||                    DRS_INTERACTIVE is not set, running on-line mode
HH:MM:SS.S -   ||
HH:MM:SS.S -   ||Validation successful. DRS installed corrected.
```

Chapter 3

Data Architecture

Described below is the file structure, after correct installation (Chapter 2).

3.1 Installed file structure

The file structure should look as follows:

```
{dir}
├── {DRS_ROOT}
│   ├── bin
│   │   └── ..... Recipes
│   ├── config
│   │   └── ..... Configuration files
│   ├── documentation
│   │   └── ..... Documentation files
│   └── SpirouDRS
│       └── ..... The DRS Module
└── {DATA_ROOT}*
    ├── calibDB
    ├── msg
    ├── raw
    │   └── YYYYMMDD ..... Observation directory
    │       └── ..... Raw observation files
    ├── reduced
    └── tmp
```

* This is the recommended file structure and raw, reduced, calibDB, msg and tmp can be changed using the {DATA_ROOT_RAW}, {DATA_ROOT_REDUCED}, {DATA_ROOT_CALIB}, {DATA_ROOT_MSG}, and {DATA_ROOT_TMP} variables in Section 2.6.

i.e. for the paths given in Section 2.6 this would be:

```
drs
├── INTROOT
│   ├── bin
│   ├── config
│   ├── documentation
│   └── SpirouDRS
└── data
    ├── calibDB
    ├── msg
    ├── raw
    │   └── YYYYMMDD
    ├── reduced
    └── tmp
```

3.2 The Installation root directory

The {INSTALL_ROOT} contains all the installed recipes, modules functions, documentation and configuration files needed to run the DRS. The file structure is set up as below:

```
{dir}
└─ {DRS_ROOT}
   ├─ bin ..................................................Recipes
   ├─ config ...............................................Configuration files
   ├─ documentation ........................................Documentation files
   └─ SpirouDRS ............................................The DRS Module
```

3.2.1 The bin directory

The bin directory is located in the {INSTALL_ROOT} directory. This contains all the recipes that can be used. A detailed description of all recipes can be found in Chapter 12 but are listed here for completeness.

- cal_DARK_spirou
- cal_DRIFT_RAW_spirou
- cal_extract_RAW_spirou
- cal_extract_RAW_spirouAB
- cal_extract_RAW_spirouC
- cal_FF_RAW_spirou
- cal_loc_RAW_spirou
- cal_SLIT_spirou
- cal_validate_spirou

3.2.2 The SPIROU module directory

The SpirouDRS directory is the SPIROU DRS package, it contains all sub-packages that contain all the worker functions and code associated with the recipes. The file structure is as follows:

```
SpirouDRS
├─ spirouBACK ..................................The SPIRou background module
├─ spirouCDB ...........................The SPIRou calibration database module
├─ spirouConfig ....................... The SPIRou configuration tools module
├─ spirouEXTOR .................................The SPIRou extraction module
├─ spirouFLAT ..................................The SPIRou Flat field module
├─ spirouImage ..................................... The SPIRou image module
├─ spirouLOCOR ................................The SPIRou localization module
├─ spirouRV ..................................The SPIRou radial velocity module
└─ spirouStartup ............................The SPIRou start up tools module
```

The modules are described in detail in Chapter 13.

3.3 The data root directory

This is the directory where all the data should be stored. The default and recommended design is to have {DATA_ROOT_RAW}, {DATA_ROOT_REDUCED}, {DATA_ROOT_CALIB}, {DATA_ROOT_MSG}, and {DATA_ROOT_TMP} as sub-directories of {DATA_ROOT}. However as in Section 2.6. these sub-directories can be defined elsewhere.

3.3.1 The raw and reduced data directories

The raw observed data is stored under the {DATA_ROOT_RAW} path, the files are stored by night in the form YYYYMMDD.

The file structure can be seen below:

```
{DATA_ROOT_RAW}
└─ YYYYMMDD ..................................................night_repository
   ├─ ..................................................Raw observation files
   ├─ dark_dark{name}.fits
   ├─ dark_flat{name}.fits
   ├─ flat_dark{name}.fits
   └─ fp_fp{name}.fits
```

3.4 The calibration database directory

```
{DATA_ROOT}
└─ calibDB or {DATA_ROOT_CALIB}
   ├─ master_calib_SPIROU.txt
   └─ ..................................................The calibration fits files
```

The calibDB contains all the calibration files that pass the quality tests and a test file 'master_calib _SPIROU.txt '. It is located at {DATA_ROOT_CALIB} or if this is not defined is located by default at the {DATA_ROOT} directory.
Each line in this file is a unique calibration file and lines are formatted in the following manner:

```text
{key} {night_repository} {filename} {human readable date} {unix time}
```

where

- {key} is a code assigned for each type of calibration file. Currently accepted keys are:
  - DARK - Created from cal_DARK_spirou
  - ORDER_PROFIL_{fiber} - Created in cal_loc_RAW_spirou
  - LOC_C - Created in cal_loc_RAW_spirou
  - TILT - Created in cal_SLIT_spirou
  - FLAT_{fiber} - Created in cal_FF_RAW_spirou
  - WAVE - Currently manually added
- {night_repository} is the raw data observation directory (in {DATA_ROOT_RAW}) normally in the form YYYYMMDD.
- {filename} is the filename of the calibration file (located in the calibDB).
- {human readable date} is the date in DD/MM/YY/HH:MM:SS.ss format taken from the header keyword 'ACQTIME1' of the file that created the calibration file.
- {unix time} is the time (as in {human readable date}) but in unix time (in seconds).

An example working master_calib_SPIROU.txt is shown below (assuming the listed files are present in {DATA_ROOT_CALIB})

```text
DARK 20170710 dark_dark02d406.fits 07/10/17/16:37:48 1499704668.0
ORDER_PROFIL_C 20170710 dark_flat02f10_order_profil_C.fits 07/10/17/17:03:50 1499706230.0
LOC_C 20170710 dark_flat02f10_loco_C.fits 07/10/17/17:03:50 1499706230.0
ORDER_PROFIL_AB 20170710 flat_dark02f10_order_profil_AB.fits 07/10/17/17:07:08 1499706428.0
LOC_AB 20170710 flat_dark02f10_loco_AB.fits 07/10/17/17:07:08 1499706428.0
TILT 20170710 fp_fp02a203_tilt.fits 07/10/17/17:25:15 1499705515.0
FLAT_C 20170710 dark_flat02f10_flat_C.fits 07/10/17/17:03:50 1499706230.0
WAVE 20170710 spirou_wave_ini3.fits 07/10/17/17:03:50 1499706230.0
```

Chapter 4

Using the DRS

There are two ways to run the DRS recipes. The first (described in Section 4.1) directly calls the code and inputs arguments (either from the command line or from python), the second way is to import the recipes in a python script and define arguments in a call to a function (see Section 4.2).

4.1 Running the DRS recipes directly

As in Chapter 2, using Linux or macOS one can run DRS recipes from the command line or from python, in windows one is required to be in python before running the scipts. Below we use cal_DARK_spirouas an example:

- To run from command line type:

CMD input

```
>> cal_DARK_spirou {YYMMDD} {Filenames})
```

- To run from python/ipython from the command line type:

CMD input

```
>> python cal_DARK_spirou {YYMMDD} {Filenames})ipython (*cal_DARK_spirou {YYMMDD} {Filenames})
```

- To run from ipython, open ipython and type:

Python/Ipython

```
run cal_DARK_spirou {YYMMDD} {Filenames})
```

4.2 Running the DRS recipes from a python script

In any operating system one can also import a recipe and call a function to run the code. This is useful in batch operations, timing tests and unit tests for example. Below we use cal_DARK_spirouas an example:

Python/Ipython

```
# import the recipe
import cal_DARK_spirou
# define the night folder name
night_name = "20170710"
# define the file(s) to run through the code
files = ['dark_dark02d406.fits']
# run code
cal_validate_spirou.main(night_name=night_name, files=files)
```

4.3 Working example of the code for SPIRou

4.3.1 Overview

For this example all files are from:

CMD input

```
>> spirou@10.102.14.81:/data/RawImages/H2RG-AT4/AT4-04/2017-07-10_15-36-18/ramps/
```

following our example data architecture (from Section 2.6 and shown explicity in Section 3.1) all files should be places in the {DATA_RAW_ROOT} (/drs/data/raw in our case).
and we will also need the current WAVE file from here:

CMD input

```
>> spirou@10.102.14.81:/data/reduced/DATA-CALIB/spirou_wave_ini3.fits
```

which needs to be placed in the {DRS_CALIB_DB} dirctory (/drs/data/calibDB in our case).
Starting with RAMP files and ending with extracted orders and calculated drifts we need to run six codes:

1. cal_DARK_spirou (See Section 12.1)
2. cal_loc_RAW_spirou(×2) (See Section 12.2)
3. cal_SLIT_spirou (See Section 12.3)
4. cal_FF_RAW_spirou(×2) (See Section 12.4)
5. (add spirou_wave_ini3.fits to calibDB)
6. cal_extract_RAW_spirouABand cal_extract_RAW_spirouC(many times) (See Section 12.5)
7. cal_DRIFT_RAW_spirou (See Section 12.6)

4.3.2 Run through from command line/python shell (Linux and macOS)

As long as all codes are excutable (see Section 2.4.3) one can run all codes from the command line or if not excutable or one has a preference for python one can run the following with 'python {command}', 'ipython {command}' or indeed through an interactive ipython session using 'run {command}'.

1. run the dark extraction on the 'dark_dark' file:

   CMD input

   ```
   >> cal_DARK_spirou.py 20170710 dark_dark02d406.fits
   ```

2. run the order localisation on the 'dark_flat' files:

   CMD input

   ```
   >> cal_loc_RAW_spirou.py 20170710 dark_flat02f10.fits dark_flat03f10.fits dark_flat04f10.fits
      dark_flat05f10.fits dark_flat06f10.fits
   ```

3. run the order localisation on the 'flat_dark' files:

   CMD input

   ```
   >> cal_loc_RAW_spirou.py 20170710 flat_dark02f10.fits flat_dark03f10.fits flat_dark04f10.fits
      flat_dark05f10.fits flat_dark06f10.fits
   ```

4. run the slit calibration on the 'fp_fp' files.

CMD input

```
>> cal_SLIT_spirou.py 20170710 fp_fp02a203.fits fp_fp03a203.fits fp_fp04a203.fits
```

5. run the flat field creation on the 'dark_flat' files:

Note: if using same files as above you will get an error message when running the file.
To solve this open the 'master_calib_SPIROU.txt ' file located in {DATA_ROOT_CALIB}.
Edit the unix date in the line that begins 'TILT' so that it is less than or equal to the unix date on rows 'ORDER_PROFIL_AB' (i.e. easiest to change it to the date on the 'ORDER_PROFIL_AB')
The human date format must match the unix date thus both must be changed if one is modified.
i.e. the 'master_calib_SPIROU.txt ' file should look go from

text

```
DARK 20170710 dark_dark02d406.fits 07/10/17/16:37:48 1499704668.0
ORDER_PROFIL_C 20170710 dark_flat02f10_order_profil_C.fits 07/10/17/17:03:50 1499706230.0
LOC_C 20170710 dark_flat02f10_loco_C.fits 07/10/17/17:03:50 1499706230.0
ORDER_PROFIL_AB 20170710 flat_dark02f10_order_profil_AB.fits 07/10/17/17:07:08
    1499706428.0
LOC_AB 20170710 flat_dark02f10_loco_AB.fits 07/10/17/17:07:08 1499706428.0
TILT 20170710 fp_fp02a203_tilt.fits 07/10/17/17:25:15 1499707515.0
```

to this:

text

```
DARK 20170710 dark_dark02d406.fits 07/10/17/16:37:48 1499704668.0
ORDER_PROFIL_C 20170710 dark_flat02f10_order_profil_C.fits 07/10/17/17:03:50 1499706230.0
LOC_C 20170710 dark_flat02f10_loco_C.fits 07/10/17/17:03:50 1499706230.0
ORDER_PROFIL_AB 20170710 flat_dark02f10_order_profil_AB.fits 07/10/17/17:07:08
    1499706428.0
LOC_AB 20170710 flat_dark02f10_loco_AB.fits 07/10/17/17:07:08 1499706428.0
TILT 20170710 fp_fp02a203_tilt.fits 07/10/17/17:07:08 1499706428.0
```

CMD input

```
>> cal_FF_RAW_spirou.py 20170710 dark_flat02f10.fits dark_flat03f10.fits dark_flat04f10.fits
   dark_flat05f10.fits dark_flat06f10.fits
```

6. Currently we do not create a new wavelength calibration file for this run. Therefore we need one (as stated in the above section). We use the one from here:

CMD input

```
>> spirou@10.102.14.81:/data/reduced/DATA-CALIB/spirou_wave_ini3.fits
```

then place it in the {DATA_ROOT_CALIB} folder. You will also need to edit the 'master_calib _SPIROU.txt ' file located in {DATA_ROOT_CALIB}.

Add the folloing line to 'master_calib_SPIROU.txt '

text

```
WAVE 20170710 spirou_wave_ini3.fits 07/10/17/17:03:50 1499706230.0
```

and the 'master_calib_SPIROU.txt' should look like this:

text

```
DARK 20170710 dark_dark02d406.fits 07/10/17/16:37:48 1499704668.0
ORDER_PROFIL_C 20170710 dark_flat02f10_order_profil_C.fits 07/10/17/17:03:50 1499706230.0
LOC_C 20170710 dark_flat02f10_loco_C.fits 07/10/17/17:03:50 1499706230.0
ORDER_PROFIL_AB 20170710 flat_dark02f10_order_profil_AB.fits 07/10/17/17:07:08 1499706428.0
LOC_AB 20170710 flat_dark02f10_loco_AB.fits 07/10/17/17:07:08 1499706428.0
TILT 20170710 fp_fp02a203_tilt.fits 07/10/17/17:07:08 1499706428.0
WAVE 20170710 spirou_wave_ini3.fits 07/10/17/17:03:50 1499706230.0
```

7. run the extraction files on the 'hcone_dark', 'dark_hcone', 'hcone_hcone', 'dark_dark_AHC1', 'hctwo_dark', 'dark_hctwo', 'hctwo-hctwo', 'dark_dark_AHC2' and 'fp_fp' files. For example for the 'fp_fp' files:

CMD input

```
>> cal_extract_RAW_spirouAB.py 20170710 fp_fp02a203.fits fp_fp03a203.fits fp_fp04a203.fits
>> cal_extract_RAW_spirouC.py 20170710 fp_fp02a203.fits fp_fp03a203.fits fp_fp04a203.fits
```

8. run the drift calculation on the 'fp_fp' files:

CMD input

```
>> @cal_DRIFT_RAW_spirou.py 20170710 @fp_fp02a203.fits fp_fp03a203.fits fp_fp04a203.fits
```

4.3.3 Run through python script

The process is in the same order as Section 4.3.2, including changing the date on the 'TILT' keyword and adding the 'WAVE' line, and adding the wave file to the calibDB folder).

Python/Ipython

```
import cal_DARK_spirou, cal_loc_RAW_spirou
import cal_SLIT_spirou, cal_FF_RAW_spirou
import cal_extract_RAW_spirou, cal_DRIFT_RAW_spirou
import matplotlib.pyplot as plt
# define constants
NIGHT_NAME = '20170710'
# cal_dark_spirou
files = ['dark_dark02d406.fits']          # set up files
cal_DARK_spirou.main(NIGHT_NAME, files)   # run cal_dark_spirou
plt.close('all')                          # close graphs
# cal_loc_RAW_spirou - flat_dark
files = ['flat_dark02f10.fits', 'flat_dark03f10.fits', 'flat_dark04f10.fits',
         'flat_dark05f10.fits','flat_dark06f10.fits']
cal_loc_RAW_spirou.main(NIGHT_NAME, files)
plt.close('all')
# cal_loc_RAW_spirou - dark_flat
files = ['dark_flat02f10.fits', 'dark_flat03f10.fits', 'dark_flat04f10.fits',
         'dark_flat05f10.fits', 'dark_flat06f10.fits']
cal_loc_RAW_spirou.main(NIGHT_NAME, files)
plt.close('all')
# cal_SLIT_spirou
files = ['fp_fp02a203.fits', 'fp_fp03a203.fits', 'fp_fp04a203.fits']
cal_SLIT_spirou.main(NIGHT_NAME, files)
plt.close('all')
# cal_FF_RAW_spirou - flat_dark
files = ['flat_dark02f10.fits', 'flat_dark03f10.fits','flat_dark04f10.fits',
         'flat_dark05f10.fits', 'flat_dark06f10.fits']
cal_FF_RAW_spirou.main(NIGHT_NAME, files)
plt.close('all')
# cal_FF_RAW_spirou - dark_flat
files = ['dark_flat02f10.fits', 'dark_flat03f10.fits', 'dark_flat04f10.fits',
         'dark_flat05f10.fits', 'dark_flat06f10.fits']
cal_FF_RAW_spirou.main(NIGHT_NAME, files)
plt.close('all')
# cal_extract_RAW_spirou - fp_fp AB
files = ['fp_fp02a203.fits', 'fp_fp03a203.fits', 'fp_fp04a203.fits']
cal_extract_RAW_spirou.main(NIGHT_NAME, files, 'AB')
plt.close('all')
# cal_extract_RAW_spirou - fp_fp C
files = ['fp_fp02a203.fits', 'fp_fp03a203.fits', 'fp_fp04a203.fits']
cal_extract_RAW_spirou.main(NIGHT_NAME, files, 'C')
plt.close('all')
# test cal_DRIFT_RAW_spirou
files = ['fp_fp02a203.fits', 'fp_fp03a203.fits', 'fp_fp04a203.fits']
cal_DRIFT_RAW_spirou.main(NIGHT_NAME, files)
plt.close('all')
```

Chapter 5

Summary of changes (AT-4)

Below we describe breifly the main differences from AT-4 build.

5.1 General

- all recipes main body of code is now in a {main()} function and this function is called in {__main__()} part of the code (the part that executes at run time). This allows recipes to be called as functions as well as being called as a standalone code or from the command line. i.e. for cal_DARK_spirou:

  Python/Ipython

  ```
  import cal_DARK_spirou

  files = ['dark_dark02d406.fits']
  night_name = '20170710'
  cal_DARK_spirou.main(night_name=night_name, files=files)
  ```

  will run the exact same procedure as:

  bash

  ```
  cal_DARK_spirou.py 201707 dark_dark02d406.fits
  ```

- {WLOG} function overhal (now in {SpirouDRS.spirouCore.spirouLog.logger()}) but aliased in most codes back to {WLOG}. This means one can use the same functionality as before:

  Python/Ipython

  ```
  WLOG("warning", "program", "message")
  ```

  However now when "error" is called an automatic exit routine is run (therefore there is no need for sys.exit after a WLOG("error", "", "") call).

- execution of pythonstartup codes removed and replaced with functions

- loading of many variables into python memory replaced with call to need dictionary object (parameter dictionary). Parameter dictionary is a custom dictionary object that as well as storing key and value pairs also can set a source for each key in the dictionary (hence the developer will always know where a variable was defined, if used correctly)

- All hard coded constants removed from running code and moved to configuration files, all variables have been described, noted their new definition locations and where they are used in the recipes and codes (see Section 10). This has allowed (and will allow) variables to either be public (i.e. in a location easily accessible by the user) or to be private (in files stored within the module). We can make many specific configuration files or a few, depending on which we deem best.

- Custom exception: {ConfigError} and {ConfigException} - designed specifically to be used with the WLOG() function

- moved core functions used in multiple recipes to sub-modules

- all plotting taken out of main codes (call to specific sub-module)

5.2 The cal_DARK_spirou recipe

- dark measurement moved to function {SpirouDRS.spirouImage.spirouImage.MeasureDark} (for clarity). This is, in part, due to the repetition of code for "Whole det", "Blue part" and "Red part".
- all plotting moved to internal functions (for clarity)
  - {SpirouDRS.spirouCore.spirouPlot.darkplot_image_and_regions} for the image/region plot
  - {SpirouDRS.spirouCore.spirouPlot.darkplot_datacut} for the DARK cutlimit plot
  - {SpirouDRS.spirouCore.spirouPlot.darkplot_histograms} for the histogram plots
- histogram plot updated, original plot plotted bin centers as a smooth peak, simple modification to make sure histogram bars are present
- writing of data is sped up by caching all HEADER keys and writing to file once with the write of the data.
- speed up
  - AT-4 v44: 4.881 seconds
  - py3: 1.890 seconds

5.3 The cal_loc_RAW_spirou recipe

- added function to convert from ADU/s to electrons {SpirouDRS.spirouImage.spirouImage.ConvertToE}
- added function to flip image {SpirouDRS.spirouImage.spirouImage.FlipImage}
- smoothed image (by a box) is now in a function (creates order_profile)
  - added different way to calculate order_profile - currently set to 'manual' be default
  - {SpirouDRS.spirouLOCOR.spirouLOCOR.BoxSmoothedImage}
  - Instead of manually working out the mean for each box you convolve the weighted image with a tophat function and the weights with a topcat function and then divide the two.
  - This gives approximately the same result (with small deviations due to the FT of a topcat function not being perfect).
  - The function can be turned back to the original manual mode by using 'mode='manual" but is slower (by a factor of $\sim \times 8$)
- added storage dictionary to store (and pass around) all variables created 'loc' - a Parameter dictionary (thus source can be set for all variables to keep track of them)
- added function to measure background and get central pixel positions {SpirouDRS.spirouLOCOR.spirouLOCOR.MeasureBkgrdGetCentPixs}
- debug plot added to plot the minimum of 'ycc' and 'ic_locseuil' {SpirouDRS.spirouCore.spirouPlot.sPlt.debug_locplot_min_ycc_loc_threshold}
- added function for locating central position (previously {SpirouDRS.spirouLOCOR.spirouLOCOR.poscolc}) - currently set to 'manual' be default
  - {SpirouDRS.spirouLOCOR.spirouLOCOR.LocateCentralOrderPositions}
  - Instead of manually working out the starts and ends of each order (with while loops) convolves a mask of cvalues > threshold with a top-hat (size=3) function such that all edges are found

  - i.e. ‘[False, True, True]‘ or ‘[True, True, False]‘ give a different value than ‘[True, True, True]‘ or ‘[False, False, False]‘ or ‘[False, False, True]‘
  - i.e. the convolution gives the sum of three elements, thus selected those elements with a sum of 2 give our edges
  - The function can be turned back to the original ’manual’ mode by using ‘mode=’manual‘ but is slower (by a factor of x2)
- debug plot added to plot the image above saturation threshold {SpirouDRS.spirouCore.spirouPlot.locplot_im_sat_threshold}
- moved ‘ctro‘,‘sigo‘,‘ac‘,‘ass‘ etc into loc (for storage and ease of use)
- the fit across each order has been split into functions
  - the initial fit is done by {SpirouDRS.spirouLOCOR.spirouLOCOR.InitialOrderFit}
  - This initial fit takes in the plotting args and thus as order is fit the fit is piped on to plot via {SpirouDRS.spirouCore.spirouPlot.locplot_order}
  - the sigma clipping fit is done by {SpirouDRS.spirouLOCOR.spirouLOCOR.SigClipOrderFit}
  - kind is used to change between ’center’ and ’fwhm’ fits (thus function is reused in both cases), kind will do the tiny bits of code which are different for each fit
  - all fit parameters are loaded into the ‘loc‘ parameter dictionary
- plot of order number against rms is move to {SpirouDRS.spirouCore.spirouPlot.locplot_order_number_against_rms}
- function created to add the 2Dlist (i.e. the coefficients) to hdict (the dictionary used to save keys to so that we only write to the fits file once)
- superimposed fit on the image is pushed into a function, this is many times faster than before - due to optimisation, {SpirouDRS.spirouLOCOR.spirouLOCOR.imageLocSuperimp}
- Writing of fits file cleaned up (header keywords written during data write)
- speed up
  - AT-4 v44: 5.697 seconds
  - py3: 2.255 seconds

5.4 The cal_SLIT_spirou recipe

- added storage dictionary to store (and pass around) all variables created ‘loc‘ - a Parameter dictionary (thus source can be set for all variables to keep track of them)
- Retrieval of coefficients from ‘_loco_‘ file moved to {SpirouDRS.spirouLOCOR.spirouLOCOR.GetCoeffs}
- Tilt finding is moved to function {SpirouDRS.spirouImage.spirouImage.GetTilt}
- Fitting the tilt is moved to function {SpirouDRS.spirouImage.spirouImage.FitTilt}
- selected order plot moved to {SpirouDRS.spirouCore.spirouPlot.slit_sorder_plot}
- slit tilt angle and fit plot moved to {SpirouDRS.spirouCore.spirouPlot.slit_tilt_angle_and_fit_plot}
- Writing of fits file cleaned up (header keywords written during data write)
- speed up
  - AT-4 v44: 11.071 seconds
  - py3: 4.386 seconds

5.5 The cal_FF_RAW_spirou recipe

- added function to replace measure_bkgr_FF, but incomplete (not currently used) would need to convert interpol.c to python (spline fitting)
- added storage dictionary to store (and pass around) all variables created 'loc' - a Parameter dictionary (thus source can be set for all variables to keep track of them)
- Created function to read TILT file from calibDB (replaces 'readkeyloco')
  - {SpirouDRS.spirouImage.spirouImage.ReadTiltFile}
  - takes in header dictionary from 'fitsfilename' in order to avoid re-opening FITS rec (acqutime used in calibDB to get max_time of calibDB entry)
- Created function to read order profile (replaces 'read_data_raw' + pre-amble)
  - {SpirouDRS.spirouImage.spirouImage.ReadOrderProfile}
  - takes in header dictionary from 'fitsfilename' in order to avoid re-opening FITS rec (acqutime used in calibDB to get max_time of calibDB entry)
- Used {SpirouDRS.spirouLOCOR.spirouLOCOR.GetCoeffs} to get the coefficients from file
- Created merge coefficients function to perform AB coefficient merge {SpirouDRS.spirouLOCOR.spirouLOCOR.MergeCoefficients}
- Updated extraction function {SpirouDRS.spirouEXTOR.spirouEXTOR.ExtracTiltWeightOrder2} - much faster as takes many of the calculations outside the pixel loop
  - i.e. calculating the pixel contribution due to tilt in array 'ww'
  - 'ww' is constant for an order, thus doesn't need to be worked out for each pixel in one order, just the multiplication between ww and the image
  - up to 8 times faster with these improvements
- 'e2ds', 'SNR', 'RMS', 'blaze' and 'flat' are stored in 'loc' parameter dictionary
- Plotting code moved to {SpirouDRS.spirouCore.spirouPlot} functions
- Writing of fits file cleaned up (header keywords written during data write)
- QC (max_signal > qc_max_signal × nbframes) moved to end, however in old code it is not used as a failure criteria so also not used to fail in new code
- speed up
  - AT-4 v44: 25.962 seconds
  - py3: 4.675 seconds

5.6 The cal_extract_RAW_spirou recipes

- Merged {cal_extract_RAW_spirouAB}, {cal_extract_RAW_spirouC} and {cal_extract_RAW_spirouALL} can still access {cal_extract_RAW_spirouAB} and {cal_extract_RAW_spirouC} but instead of being modified copies of the code they are just wrappers for {cal_extract_RAW_spirou.py} (i.e. they forward the fiber type)
- added storage dictionary to store (and pass around) all variables created 'loc' - a Parameter dictionary (thus source can be set for all variables to keep track of them)

- Created function to read TILT file from calibDB (replaces ‘readkeyloco‘)
  - {SpirouDRS.spirouImage.spirouImage.ReadTiltFile}
  - takes in header dictionary from ‘fitsfilename‘ in order to avoid re-opening FITS rec (acqutime used in calibDB to get max_time of calibDB entry)
- Created function to read WAVE file from calibDB (replaces ‘read_data_raw(‘)
  - {SpirouDRS.spirouImage.spirouImage.ReadWaveFile}
  - takes in header dictionary from ‘fitsfilename‘ in order to avoid re-opening FITS rec (acqutime used in calibDB to get max_time of calibDB entry)
- Used {SpirouDRS.spirouLOCOR.spirouLOCOR.GetCoeffs} to get the coefficients from file
- Created function to read order profile (replaces ‘read_data_raw‘ + pre-amble)
  - {SpirouDRS.spirouImage.spirouImage.ReadOrderProfile}
  - takes in header dictionary from ‘fitsfilename‘ in order to avoid re-opening FITS rec (acqutime used in calibDB to get max_time of calibDB entry)
- Created merge coefficients function to perform AB coefficient merge {SpirouDRS.spirouLOCOR.spirouLOCOR.MergeCoefficients}
- New structures above replace the need for specific fiber sections (’AB’, ’C’, ’A’, ’B’) (In {cal_extract_RAW_spirouALL} and individual setups for {cal_extract_RAW_spirouAB} and {cal_extract_RAW_spirouC})
- all extraction functions passed into {spirouEXTOR} to wrapper functions ({SpirouDRS.spirouEXTOR.spirouEXTOR.ExtractOrder}, {SpirouDRS.spirouEXTOR.spirouEXTOR.ExtractTiltOrder}, {SpirouDRS.spirouEXTOR.spirouEXTOR.ExtractTiltWeightOrder} and {SpirouDRS.spirouEXTOR.spirouEXTOR.ExtractWeightOrder}) these are then run into {SpirouDRS.spirouEXTOR.spirouEXTOR.ExtractionWrapper} and processed accordingly
- Added a timing string (to record timings of all extraction processes) use ‘print(timing))‘ to view
- ‘e2ds‘ and ‘SNR‘ stored in ‘loc‘
- Plotting code moved to {SpirouDRS.spirouCore.spirouPlot} functions
- Writing of fits file cleaned up (header keywords written during data write)
- QC (max_signal > qc_max_signal × nbframes) moved to end, however in old code it is not used as a failure criteria so also not used to fail in new code
- speed up
  - AT-4 v44: 60.852
  - py3: 8.694
  - Extraction timing Py3:
    - ExtractOrder = 0.025 s
    - ExtractTiltOrder = 0.060 s
    - ExtractTiltWeightOrder = 0.141 s
    - ExtractWeightOrder = 0.070 s
  - Extraction timing AT-4 v46:
    - ExtractOrder (Fortran) = 0.019 s

    - ExtractOrder (Py2) = 0.085 s
    - ExtractTiltOrder = 0.766 s
    - ExtractTiltWeightOrder = 0.840 s
    - ExtractWeightOrder = 0.156 s
  - Speed increase (Py3 over AT-4 v46)
    - ExtractOrder (Py3 → Fortran) = slower x 1.3 times slower
    - ExtractOrder (Py3 → Py) = faster x 3.4 times faster
    - ExtractTiltOrder (Py3 → Py = faster x12.9 times faster
    - ExtractTiltWeightOrder (Py3 → Py) = faster x6.0 times faster
    - ExtractWeightOrder (Py3 → Py) = faster x2.2 times faster

5.7 The cal_DRIFT_RAW_spirou recipe

- acqtime (bjdref) got from header using {SpirouDRS.spirouImage.spirouImage.GetAcqTime}
  - can be used to get both 'human' readible and 'unix' time (use key kind='human' or kind='unix)
- Created function to read TILT file from calibDB (replaces 'readkeyloco')
  - {SpirouDRS.spirouImage.spirouImage.ReadTiltFile}
  - takes in header dictionary from 'fitsfilename' in order to avoid re-opening FITS rec (acqutime used in calibDB to get max_time of calibDB entry)
- Created function to read WAVE file from calibDB (replaces 'read_data_raw(')
  - {SpirouDRS.spirouImage.spirouImage.ReadWaveFile}
  - takes in header dictionary from 'fitsfilename' in order to avoid re-opening FITS rec (acqutime used in calibDB to get max_time of calibDB entry)
- Used {SpirouDRS.spirouLOCOR.spirouLOCOR.GetCoeffs} to get the coefficients from file
- Created function to read order profile (replaces 'read_data_raw' + pre-amble)
  - {SpirouDRS.spirouImage.spirouImage.ReadOrderProfile}
  - takes in header dictionary from 'fitsfilename' in order to avoid re-opening FITS rec (acqutime used in calibDB to get max_time of calibDB entry)
- new extraction (see cal_extract_RAW_spirouabove).
- delta RV RMS calculation in {SpirouDRS.spirouRV.spirouRV.DeltaVrms2D}
  - where arguments are 'speref' and 'wave' (stored in 'loc')
  - where keyword arguments are 'sigdet', 'size' and 'threshold' (stored in p)
- all functionality to do with listing files moved to {SpirouDRS.spirouImage.spirouImage.GetAllSimilarFiles} - no need for "alphanumeric short"/"nice sort" - 'np.sort(x)' does this
- Renormlisation and cosmics correction in {SpirouDRS.spirouRV.spirouRV.ReNormCosmic2D}
  - where arguments are 'speref' and 'spe' (stored in 'loc')
  - where keyword arguments are 'cut', 'size' and 'threshold' (stored in p)
- RV drift calculated

  - {SpirouDRS.spirouRV.spirouRV.CalcRVdrift2D}
  - where arguments are ‘speref‘, ‘spen‘ and ‘wave‘ (‘speref‘ and ‘spen‘ stored in loc)
  - where keyword arguments are ‘sigdet‘, ‘size‘ and ‘threshold‘ (stored in p)
- added an option (drift_type_e2ds) to decide between getting drift using a weighted mean or using a median (to combine all orders)
- ‘drift‘, ‘errdrift‘, ‘deltatime‘, ‘mdrift‘, ‘merrdrift‘ stored in loc
- Writing of fits file cleaned up (header keywords written during data write)
- speed up
  - AT-4 v44: 22.556 s
  - py3: 8.143 s

5.8 The cal_BADPIX_spirou recipe

- loading of custom arguments moved to {SpirouDRS.spirouStartup.spirouStartup.GetCustomFromRuntime}
- loading of files moved to {SpirouDRS.spirouImage.spirouImage.ReadImage}
- normalising flat and median of flat moved to {SpirouDRS.spirouImage.spirouImage.NormMedianFlat}
- locating bad pixels moved to {SpirouDRS.spirouImage.spirouImage.LocateBadPixels}
- instead of taking the 90th pixel in flattened meadian flat image now work out the 90th percentile of finite values (will lead to a slightly more correct normalisation value)
- Writing of fits file cleaned up (header keywords written during data write)

5.9 The cal_DRIFT-E2DS_spirou recipe

- loading of custom arguments for reference file
- acqtime (bjdref) got from header using {SpirouDRS.spirouImage.spirouImage.GetAcqTime}
  - can be used to get both ‘human‘ readible and ‘unix‘ time (use key kind=‘human‘ or kind=‘unix)
- Created function to read TILT file from calibDB (replaces ‘readkeyloco‘)
  - {SpirouDRS.spirouImage.spirouImage.ReadTiltFile}
  - takes in header dictionary from ‘fitsfilename‘ in order to avoid re-opening FITS rec (acqutime used in calibDB to get max_time of calibDB entry)
- Created function to read WAVE file from calibDB (replaces ‘read_data_raw(‘)
  - {SpirouDRS.spirouImage.spirouImage.ReadWaveFile}
  - takes in header dictionary from ‘fitsfilename‘ in order to avoid re-opening FITS rec (acqutime used in calibDB to get max_time of calibDB entry)
- delta RV RMS calculation in {SpirouDRS.spirouRV.spirouRV.DeltaVrms2D}
  - where arguments are ‘speref‘ and ‘wave‘ (stored in ‘loc‘)
  - where keyword arguments are ‘sigdet‘, ‘size‘ and ‘threshold‘ (stored in p)

- all functionality to do with listing files moved to {SpirouDRS.spirouImage.spirouImage.GetAllSimilarFiles} - no need for "alphanumeric short"/"nice sort" - ‘np.sort(x)‘ does this
- Renormlisation and cosmics correction in {SpirouDRS.spirouRV.spirouRV.ReNormCosmic2D}
  - where arguments are ‘speref‘ and ‘spe‘ (stored in ‘loc‘)
  - where keyword arguments are ‘cut‘, ‘size‘ and ‘threshold‘ (stored in p)
- RV drift calculated
  - {SpirouDRS.spirouRV.spirouRV.CalcRVdrift2D}
  - where arguments are ‘speref‘, ‘spen‘ and ‘wave‘ (‘speref‘ and ‘spen‘ stored in loc)
  - where keyword arguments are ‘sigdet‘, ‘size‘ and ‘threshold‘ (stored in p)
- added an option (drift_type_e2ds) to decide between getting drift using a weighted mean or using a median (to combine all orders)
- ‘drift‘, ‘errdrift‘, ‘deltatime‘, ‘mdrift‘, ‘merrdrift‘ stored in loc
- Writing of fits file cleaned up (header keywords written during data write)
- new functions to save to .tbl format (SpirouDRS.spirouImage.spirouImage.MakeTable and SpirouDRS .spirouImage.spirouImage.WriteTable)

5.10 The cal_DRIFT-PEAK_E2DS_spirou recipe

- loading of custom arguments for reference file
- acqtime (bjdref) got from header using {SpirouDRS.spirouImage.spirouImage.GetAcqTime}
  - can be used to get both ‘human‘ readible and ‘unix‘ time (use key kind=‘human‘ or kind=‘unix)
- Created function to read WAVE file from calibDB (replaces ‘read_data_raw(‘)
  - {SpirouDRS.spirouImage.spirouImage.ReadWaveFile}
  - takes in header dictionary from ‘fitsfilename‘ in order to avoid re-opening FITS rec (acqutime used in calibDB to get max_time of calibDB entry)
- FP identification moved to {SpirouDRS.spirouRV.spirouRV.CreateDriftFile()}
- Removal of wide peaks moved to {SpirouDRS.spirouRV.spirouRV.RemoveWidePeaks()}
- Drift calcualtion moved to {SpirouDRS.spirouRV.spirouRV.GetDrift()}
- Removal of zero drifts moved to {SpirouDRS.spirouRV.spirouRV.RemoveZeroPeaks()}
- all functionality to do with listing files moved to {SpirouDRS.spirouImage.spirouImage.GetAllSimilarFiles} - no need for "alphanumeric short"/"nice sort" - ‘np.sort(x)‘ does this
- Pearson R test moved to {SpirouDRS.spirouRV.spirouRV.PearsonRtest()}
- Sigma clipping moved to {SpirouDRS.spirouRV.spirouRV.SigmaClip()}
- Drift calculation moved to {SpirouDRS.spirouRV.spirouRV.DriftPerOrder()} (for per order drifts) and {SpirouDRS.spirouRV.spirouRV.DriftAllOrders()} (for drift per file)
- Writing of fits file cleaned up (header keywords written during data write)
- new functions to save to .tbl format (SpirouDRS.spirouImage.spirouImage.MakeTable and SpirouDRS .spirouImage.spirouImage.WriteTable)

Chapter 6

Current to do list

Below is the current to do list and any things that need to be addressed before release.

6.1 General

- Write codes (see sections below)
- Write unit tests (see sections below)
- Write documentation (see section below)
- Doc strings for all modules, functions and function aliases
- Need to sort out public and private variables (and keywords)
- Need to move user configuration file to, for example, /home/$user/.spirou_config and call the default values from a private location
- Check confirugation variable values are valid at startup of recipes (avoids crashes later)
- Can we remove 'special_config_SPIROU' configuration file call as the file does not exist?
- fitsfilename is the last file in a group of files - is this correct or should it be the first (as initially defined)?
- 'nbcos' in {SpirouDRS.spirouImage.spirouImage} is not used - what is it?
- 'image_gap' in {SpirouDRS.spirouLOCOR.spirouLOCOR} is set to zero, what is this?
- Some keywords added to header but not updated in any recipe - should they be removed (or updated)?
- Write a setup.py installer / checker for prerequisites (Last step)

6.2 Documentation

- Write introduction (leading paragraph) ..... Dev
- ~~Write installation~~ ..... User + Dev
- ~~Write data architecture~~ ..... User + Dev
- ~~Write using the DRS~~ ..... User + Dev
- ~~Write summary of changes~~ ..... Dev
- ~~Write Todo chapter~~ ..... Dev
- ~~Write coding style chapter~~ ..... Dev
- ~~Write documentation chapter~~ ..... Dev
- ~~Write input keywords chapter~~ ..... Dev
- ~~Write variables chapter~~ ..... User + Dev

- ~~Write output keywords chapter~~ ..... Dev
- Write recipes chapter ..... User + Dev
- Write module chapter ..... Dev

6.3 The cal_DARK_spirou recipe

- ~~convert code from AT-4 v43 to run on python 3~~
- ~~add variables and keywords to documentation~~
- ~~Update from AT-4 v43 to current~~
- Unit test - comparing outputs this version to AT-4

6.4 The cal_loc_RAW_spirou recipe

- ~~convert code from AT-4 v43 to run on python 3~~
- ~~add variables and keywords to documentation~~
- ~~Update from AT-4 v43 to current~~
- Unit test - comparing outputs this version to AT-4

6.5 The cal_SLIT_spirou recipe

- ~~convert code from AT-4 v43 to run on python 3~~
- ~~add variables and keywords to documentation~~
- ~~Update from AT-4 v43 to current~~
- Unit test - comparing outputs this version to AT-4

6.6 The cal_FF_RAW_spirou recipe

- ~~convert code from AT-4 v43 to run on python 3~~
- ~~add variables and keywords to documentation~~
- {SpirouDRS.spirouBACK.spirouBACK.measure_background_flatfield()} needs converting from C to python (interpol.c) - currently not used so not converted – background set to zero.
- {SpirouDRS.spirouBACK.spirouBACK.measure_min_max()} why is the max_signal the third biggest value and not a percentile?
- Quality control test {QC_MAX_SIGNAL} ignored for some reason - Why?
- ~~Update from AT-4 v43 to current~~
- Unit test - comparing outputs this version to AT-4

6.7 The cal_extract_RAW_spirou recipes

- ~~convert code from AT-4 v43 to run on python 3~~
- ~~add variables and keywords to documentation~~
- {SpirouDRS.spirouBACK.spirouBACK.measure_background_flatfield()} needs converting from C to python (interpol.c) - currently not used so not converted – background set to zero.
- {SpirouDRS.spirouBACK.spirouBACK.measure_min_max()} why is the max_signal the third biggest value and not a percentile?
- Quality control test {QC_MAX_SIGNAL} ignored for some reason - Why?
- Inconsitency in adding "lower" and "upper" contribution due to pixel rounding in pixel extraction process.
- ~~Update from AT-4 v43 to current~~
- Unit test - comparing outputs this version to AT-4

6.8 The cal_DRIFT_RAW_spirou recipe

- ~~convert code from AT-4 v43 to run on python 3~~
- ~~add variables and keywords to documentation~~
- ~~Update from AT-4 v43 to current~~
- Unit test - comparing outputs this version to AT-4

6.9 The cal_BADPIX_spirou recipe

- ~~convert code from AT-4 to run on python 3~~
- ~~add variables and keywords to documentation~~
- Unit test - comparing outputs this version to AT-4

6.10 The cal_DRIFT_E2DS_spirou recipe

- ~~convert code from AT-4 to run on python 3~~
- ~~add variables and keywords to documentation~~
- ~~Update from AT-4 v43 to current~~
- Unit test - comparing outputs this version to AT-4

6.11 The cal_DRIFT-PEAK_E2DS_spirou recipe

- ~~convert code from AT-4 to run on python 3~~
- add variables and keywords to documentation
- Unit test - comparing outputs this version to AT-4

6.12 The cal_HC_E2DS_spirou recipe

- convert code from AT-4 to run on python 3
- add variables and keywords to documentation
- Unit test - comparing outputs this version to AT-4

6.13 The cal_WAVE_E2DS_spirou recipe

- convert code from AT-4 to run on python 3
- add variables and keywords to documentation
- Unit test - comparing outputs this version to AT-4

6.14 The cal_CCF_E2DS_spirou recipe

- convert code from AT-4 to run on python 3
- add variables and keywords to documentation
- Unit test - comparing outputs this version to AT-4

6.15 The pol_spirou recipe

- convert code from AT-4 to run on python 3
- add variables and keywords to documentation
- Unit test - comparing outputs this version to AT-4

Chapter 7

Coding style and standardization

To keep the code neat, tidy, consistent and professional the following sections suggest guideline by which the DRS should conform to.

7.1 PEP 8 - A style guide for python code

PEP 8 is a style guide for python it lays out a specific way to format python code, a full guide can be found here: https://www.python.org/dev/peps/pep-0008/ but the following summarizes the main points used in the DRS.

- Code lay-out
  - 4 spaces per indentation level (spaces not tabs)
  - Continuation lines should align wrapped elements
  - Maximum line length of 79 characters
  - Surround top-level functions and class definitions with two blank lines (methods with one blank line and all other code with one blank line maximum)
  - imports should usually be on separate lines
- Whitespace in expressions and statements
  - No white spaces immediately inside parentheses, brackets or braces
  - No white spaces immediately before a comma, semicolon, or colon (exception for slicing)
  - No white spaces immediately before the open parenthesis that starts the argument list of a function call
  - No white spaces immediately before the open parenthesis that starts an indexing or slicing
  - Exactly one white space around an assignment (or other) operator
  - No space around the = sign when used to indicate a keyword argument or a default parameter value
  - Avoid compound statements (multiple statements on the same line)
- Comments
  - Comments should start with a # and be followed by a single white space
  - In-line comments should be used sparingly
  - All functions, classes and methods should have a valid document string (see here: https://www.python.org/dev/peps/pep-0257)
- Naming conventions
  - Never use lowercase letter el 'l', uppercase letter 'oh' 'O', or uppercase letter 'eye' 'I' as single character variables names
  - Class Names should normally use CamelCase (words should be Capitalized)
  - Functions names should be lowercase with words separated by underscores as necessary (same is true for global variable names)
  - Constants defined on a module level should be written in capital letters with underscores separating words

7.2 DRS specific style and standardization

In addition to PEP-8 we stick to some extra style and standardization points, these include some custom objects to help the ease of development and user experience.

7.2.1 Functions from sub-modules

Unlike 'normal' functions these are written in CamelCase without underscores between words. This is done to distinguish them from standard functions. They are always defined in a module (or sub-modules) __init__() code and are essentially public aliases to module level code. An example is presented below.

Python/Ipython

```
# --------------------------
# in the module file spirouMath.py
# --------------------------
def add_x_to_y(x, y):
  """
  Returns the summation of x and y
  :param x: float, the first term to add
  :param y: float, the second term to add
  :return z: float, the summation of x and y
  """
  # add x to y
  z = x + y
  # return z
  return z

# --------------------------
# in the __init__ file for spirouCore
# --------------------------
# import from local code
from . import spirouMath
# publicly defined alias to local code function
AddXtoY = spirouMath.add_x_to_y

# --------------------------
# in the recipe
# --------------------------
# import sub-module
from SpirouDRS import spirouCore
# set up constants
x = 4.123
y = 5.234
# add via function
z = spirouCore.AddXtoY(x, y)
```

7.2.2 The logger (WLOG)

As in previous version of the DRS the printing and logging is controlled by a function. In this version of the DRS this is in {SpirouDRS.spirouCore.spirouLog.logger} but in most recipes/modules this is aliased to {WLOG}. The {WLOG} function controls both the printing to the screen (standard output) and to a log file. Where and how this is done is controlled by several variables.

The format of the log entry (whether it is printed to the standard output or to the logging file) is as follows:

Python/Ipython

```
WLOG(level, program, message)
```

and produces the following entry (in log or standard output)

Command line output

```
HH:MM:SS.s - char | program | message
```

where the 'char' is dependent on the input level.
The 'char' and level are a dictionary pair in the form 'level = char' and is controlled by {trig_key} (see section 10.19) i.e. by default the level char pairs are:

Python/Ipython

```
dict(all=' ', error='!', warning='@', info='*', graph='~')
```

The level also determines whether or not a message is shown in the screen (standard output) or in the log. A log message will be shown if it has a numeric value (defined in {write_level}) higher than that set in {PRINT_LEVEL} for printing to the screen (standard output) or set in {LOG_LEVEL} for printing to the log.

i.e.:

Python/Ipython

```
write_level = dict(error=3, warning=2, info=1)
trig_key = dict(all=' ', error='!', warning='@', info='*', graph='~')
PRINT_LEVEL = 'warning'

WLOG('info', 'program', 'Info message')
WLOG('warning', 'program', 'Warning message')
WLOG('error', 'program', 'Error message')
```

returns

Command line output

```
HH:MM:SS.s - @ |program|Warning message
HH:MM:SS.s - ! |program|Error message
```

> **Note:** Note the info message was not shown as info=1 and {PRINT_LEVEL} is set to warning=2.

In addition to logging the certain levels can be set to exit the DRS recipe when they are used. They are defined in {exit_levels} and exiting python is controlled via {exit} and {Logger exit type}.

i.e.

Python/Ipython

```
write_level = dict(error=3, warning=2, info=1)
trig_key = dict(all=' ', error='!', warning='@', info='*', graph='~')
exit_levels = ['error']
PRINT_LEVEL = 'warning'

WLOG('error', 'program', 'Error message')
WLOG('info', 'program', 'Info message')
WLOG('warning', 'program', 'Warning message')
```

returns

Command line output

```
HH:MM:SS.s - ! |program|Error message
```

Note: Note that 'WLOG('error')' triggered the recipe/module to exit python, thus no other logs were printed.

7.2.3 The Parameter Dictionary Object

While running the DRS there are many variables defined in many places that are used throughout the recipes, DRS module and sub-modules, defined in configuration files and from certain sub-modules and recipes. It is important as a developer (and for proper error handling) to keep track of where this variables are being defined and changed in the DRS.

For this reason, and for convenience for passing between functions and recipes, a new object, based on a dictionary has been defined to handle all variables defined throughout the DRS. This is the parameter dictionary ({ParamDict}) class (defined in {SpirouDRS.spirouConfig.spirouConfig}).

The {ParamDict}is a custom dictionary class (that inherits all attributes and methods from the standard python dictionary object), with the ability to get and set a source for each key value pair. In addition to this all variables stored are **insensitive to case** (i.e. uppercase variables, lowercase variables and mixed case variables are stored as the **same** variable).

Construct/initiate the {ParamDict}in the same way one would a python dictionary:

Python/Ipython

```
# as an empty dictionary
p1 = ParamDict()
# from a list of keys and values (using zip)
p2 = ParamDict(zip(keys, values))
```

Once created key, value pairs are created the same way one would with a python dictionary.

Python/Ipython

```
# set a key, value pair
p1['test'] = 1
# ParamDict are case insensitive 'Test' overwrites 'test' and 'teST'
p1['Test'] = 99
```

After creating a key the source should be set. This can be done as follows:

Python/Ipython

```
# -----------------------
# Set a single source
# -----------------------
# set the key value pair
p1['test'] = 1
# set the source
p1.set_source('test', 'test.py/__main__()')
```

One can also add a set of sources (after creating multiple key value pairs)

Python/Ipython

```
# -----------------------
# Set a list of sources
# -----------------------
# set the key value pairs
p1['a'] = 1
p1['b'] = 2
p1['c'] = 3
# set the sources
p1.set_sources(['a', 'b', 'c'], 'SpirouConfig.DefineConstants()')
```

or one can set all sources in the {ParamDict}to a specific source

Note: Note set all sources will change every source in the {ParamDict}so should only be used after {ParamDict}created from a set of key value pairs

Python/Ipython

```
# -----------------------
# Set all sources
# -----------------------
# create ParamDict
keys = ['a', 'b', 'c', 'd', 'e']
values = [1, 2, 3, 4, 5]
p3 = ParamDict(zip(keys, values))
# set all sources
p3.set_all_sources('SpirouMath.LetterNumbers()')
```

7.2.4 Configuration Error and Exception

As mentioned above in section 7.2.3 it is important to handle errors caused by variable definition. Included in the parameter dictionary definitions are a new set of exception handlers to be used with {ParamDict}and the {SpirouDRS.spirouCore.spirouLog.logger} (aliased to {WLOG} in most modules/recipes). It is very similar to standard python Exceptions but adds some new methods that can be accessed to be used with {WLOG}.

An example is below of the {ConfigError} exception (without using {ParamDict})

Python/Ipython

```
def a_function():
    try:
        # some_code that causes an exception
        x = dict()
        y = x['a']
        return y
    except KeyError:
        # define a log message
        message = 'a was not found in dictionary x'
        raise ConfigError(message, level='error')

# Main code:
try:
    a_function()
except ConfigError as e:
    WLOG(e.level, 'program', e.message)
```

This functionality is coded into {ParamDict}(with a {WLOG} level set to 'error') thus one only needs the following code:

Python/Ipython

```
# set up the ParamDict
x = ParamDict()
# Main code:
try:
    y = x['add']
except ConfigError as e:
    WLOG(e.level, 'program', e.message)
```

and the result will be as follows:

Command line output

```
HH:MM:SS.s - ! |program|Parameter "add" not found in parameter dictionary
```

Note: Due to WLOG 'error' currently meaning the code is exited a missing parameter will print the above message and then exit using the {log_exit_type} exit strategy (see section 7.2.2).

Chapter 8

Writing the documentation

8.1 Documentation Architecture

The documentation is written in LaTeXand to ease writing many packages and customizations are used. The documentation is located in the ./documentation folder. Both the user documentation and the developer documentation (this document) are written together.

The main .tex files are User_guide_spirou_drs.tex and Dev_guide_spirou_drs.tex these are the files that should be compiled by LaTeX. As well as this file there are four directories, the 'Chapters' directory (containing the content of each chaper), the 'Config' directory (containing custom commands, formatting and constants - see Section 8.4, Section 8.5, and Section 8.6), the 'Figures' directory (containing all figures and graphics) and the 'Tables' directory containing table .tex files.

The documentation is currently written using the 'memoir' class file (texdoc.net/texmf-dist/doc/latex/memoir/memman.pdf) and uses custom chapter styles from ctan.org/pkg/memoirchapterstyles (Contained within the .documentation/Config/preamble.tex file).

8.2 Required LaTeXpackages

To compile in LaTeXone needs the following document class:

- memoir ..... Typeset fiction, non-fiction and mathematical books

To compile in LaTeXone needs the following packages:

- inputenc ..... utf8 encoding
- fontenc ..... Standard package for selecting font encodings
- babel ..... Multilingual support for Plain TeXor LaTeX
- microtype ..... Subliminal refinements towards typographical perfection
- amsmath ..... AMS mathematical facilities for LaTeX
- amssymb ..... AMS symbols for LaTeX
- mathtools ..... Mathematical tools to use with amsmath
- memhfixc ..... Adjustment for using hyperref in memoir documents
- graphicx ..... Enhanced support for graphics
- listings ..... Typeset source code listings using LaTeX
- xcolor ..... Driver-independent color extensions for LaTeXand pdfLaTeX
- hyperref ..... Extensive support for hypertext in LaTeX
- dirtree ..... Display trees in the style of windows explorer
- framed ..... Framed or shaded regions that can break across pages
- multirow ..... Create tabular cells spanning multiple rows
- float ..... Improved interface for floating objects

- background ..... Placement of background material on pages of a document
- tcolorbox ..... Coloured boxes, for LaTeXexamples and theorems
- eso-pic ..... Add picture commands (or backgrounds) to every page
- ulem ..... Package for underlining

8.3 Developer documentation content

As mentioned above we write the developer documentation and user guide using the same files, for this reason one needs a way to distinguish content that is unique to the user documentation or the developer documentation. This is done using the boolean statement '\ifdevguide' (defined in the main .tex files for the user documentation - \devguidefalse - and developer documentation \devguidetrue).
An example of a different content for each type of documentation is below:

LaTeX
```
\ifdevguide
This is the developer guide.
\else
This is the user guide.
\fi
```

This is the developer guide.

an example of content only for the developer guide is below:

LaTeX
```
\ifdevguide
This section is only for developers
\fi
```

This section is only for developers

an example of content only for the user guide is below:

LaTeX
```
\ifdevguide
\else
This section is only for developers
\fi
```

Note: As this is the developer guide the content for the user guide only will not be present.

Note: It is probably never the case where the user documentation will have content that the developer documentation does not need.

8.4 Custom Commands

To ease writing the documentation some custom commands are defined in ./documentation/Config/commands.tex. These include the following:

- \definevariable - used to create in-text variables i.e. {VARIABLE}. i.e.:

LaTeX
```
The variable \definevariable{VARIABLE} can be used.
```

The variable {VARIABLE} can be used.

- \definekeyword - used to create in-text keywords i.e. {KEYWORD}. i.e.:

LaTeX
```
The keyword \definekeyword{KEYWORD} can be used.
```

The keyword {KEYWORD} can be used.

- \Program - used to highligh a program (writen in small caps). i.e.:

LaTeX
```
The program \Program{AstroPy} can be used.
```

The program AstroPy can be used.

- \ParameterEntry - used to define a parameter entry. It requires 8 arguments (Variable title, Description, variable name, default value, which recipe it is used in, the place the variable is defined, the code/module/function it is used in – dev only, and the visibility level – dev only). i.e.:

LaTeX

```
\ParameterEntry{Variable title}
{Description of the variable}
{VARIABLE\_NAME}
{Default Value}{The recipe used the variable is used in.}
{The place where the variable is defined.}
{The code (module + function) where variable is used.}
{
Who should be able to change this variable, levels are as follows:
\begin{itemize}
        \item Public: Everyone (including the user)
        \item Private: Only the developer
\end{itemize}
}
```

Variable title (VARIABLE_NAME)

Description of the variable

VARIABLE_NAME = Default Value

Used in:	The recipe used the variable is used in.
Defined in:	The place where the variable is defined.
Called in:	The code (module + function) where variable is used.
Level:	Who should be able to change this variable, levels are as follows:

- Public: Everyone (including the user)
- Private: Only the developer

- \PseudoParamEntry - used to define a pseudo parameter entry. It requires 6 arguments (Variable title, Description, variable name, which recipe it is used in, the place the variable is defined, the code/module/function it is used in). It is only available for the developer guide. i.e.:

LaTeX

```
\PseudoParamEntry{Variable title}
{Description of the variable}
{VARIABLE\_NAME}
{The recipe used the variable is used in.}
{The place where the variable is defined.}
{The code (module + function) where variable is used.}
```

Variable title (VARIABLE_NAME)

Description of the variable

VARIABLE_NAME

Used in:	The recipe used the variable is used in.
Defined in:	The place where the variable is defined.
Called in:	The code (module + function) where variable is used.

Note: \PseudoParamEntry is identical to \ParameterEntry other than not requiring a default value and not requiring a visibility level (as it is generally used for code thus a simple value can not be given cleanly and will always be a private variable).

- \KeywordEntry - used to define a keyword entry. It requires 9 arguments (Keyword title, Description, keyword name, HEADER key name, default HEADER value, HEADER comment, which recipe it is used in, the place the variable is defined, the code/module/function it is used in). It is only available for the developer guide.

LaTeX

```
\KeywordEntry{Keyword title}
{Description of the keyword}
{kw\_variable}{HEADER key}
{Default HEADER value}{HEADER comment}
{The recipe the keyword is used in}
{The place where the keyword is defined}
{The code where the keyword is used.}
```

Keyword title (kw_variable)

Description of the keyword

kw_variable = ["HEADER key", "Default HEADER value", "HEADER comment"]

HEADER file entry:

HEADER key = Default HEADER value \ HEADER comment

Used in: The recipe the keyword is used in
Defined in: The place where the keyword is defined
Called in: The code where the keyword is used.

- \customdirtree - used to create a directory tree, so add background use with the tcustomdir environment (see 8.6). The format of each line is

```text
.{level} {directory}.
```

each line must start with a period and end with a period, comments can be added using the \DTcomment command.
an example is shown below:

LaTeX

```
        The file structure should look as follows:
\begin{tcustomdir}
\customdirtree{%
.1 home.
.2 user1.
.3 Downloads\DTcomment{User 1 downloads}.
.3 Documents.
.4 \DTcomment{Many documents in here}.
.2 user2.
.3 Downloads.
.3 Documents\DTcomment{User 2 documents}.
}
\end{tcustomdir}
```

The file structure should look as follows:

```
home
├── user1
│   ├── Downloads ..... User 1 downloads
│   └── Documents
│       └── ..... Many documents in here
└── user2
    ├── Downloads
    └── Documents ..... User 2 documents
```

8.5 Constants

Many constants are setup to ease the writing of this documentation. These can be found in the ./documentation/Config/constants.tex file. These are defined and use in the form:

LaTeX
```
% define constant
\newcommand{\ConstantName}{ConstantName}
% user constant
The constant is called \ConstantName
```

The constant is called ConstantName

Please check out the constants.tex file for the list of which constanats are currently defined.

8.6 Code formatting

This section deals with the textbox, cmdbox, pythonbox and LaTeXboxes seen throughout the documentation. These are defined in ./documentation/Config/code_format.tex along with many style definitions (that only need to be changed to change colours/box styles) - this is left out of this guide for berivity.

- A line/lines of text (that are to be edited in a text editor):

LaTeX
```
\begin{textbox}
<# A variable name that can be changes to a specific value>
@VARIABLE_NAME@ = "Variable Value"
\end{textbox}
```

text
```
# A variable name that can be changes to a specific value
VARIABLE_NAME = "Variable Value"
```

- A line/lines of text (that are to be edited in bash):

LaTeX
```
\begin{bashbox}
#!/usr/bin/bash
# Find out which console you are using
echo $0
# Set environment Hello
export Hello="Hello"
\end{bashbox}
```

bash
```
#!/usr/bin/bash
# Find out which console you are using
echo $0
# Set environment Hello
export Hello="Hello"
```

- A line/lines of text (that are to be edited in bash):

LaTeX

```
\begin{cshbox}
#!/usr/bin/tcsh
# Find out which console you are using
echo $0
# Set environment Hello
setenv Hello "Hello"
\end{cshbox}
```

tcsh/csh

```
#!/usr/bin/tcsh
# Find out which console you are using
echo $0
# Set environment Hello
setenv Hello "Hello"
```

- A line/lines of text to be run in the command shell:

LaTeX

```
\begin{cmdbox}
cd (*$\sim$*)/Downloads
\end{cmdbox}
```

CMD input

```
>> cd ~/Downloads
```

- A line/lines of text that is print out to the screen (standard output):

LaTeX

```
\begin{cmdboxprint}
 This is a print out in the command line
 produced by using the echo command
\end{cmdboxprint}
```

Command line output

```
This is a print out in the command line
produced by using the echo command
```

- A line/lines of text in the python terminal or python script

LaTeX
```
\begin{pythonbox}
import numpy as np
print("Hello world")
print("{0} seconds".format(np.sqrt(25)))
\end{pythonbox}
```

Python/Ipython
```
import numpy as np
print("Hello world")
print("{0} seconds".format(np.sqrt(25)))
```

- A line/lines of text in LaTeXcode (in raw form and then compiled form).

LaTeX
```
\begin{latexbox}
This is my \LaTeX code.
\end{latexbox}
```

LaTeX
```
This is my \LaTeX code.
```

This is my LaTeXcode.

- highlighted textbox:

LaTeX
```
\begin{thighlight}
Highlighted section
\end{thighlight}
```

Highlighted section

- Custom directory tree (highlighted section):

LaTeX
```
\begin{tcustomdir}
\customdirtree{%
.1 home.
.2 \DTcomment{User directories here}.
}
\end{tcustomdir}
```

home
└── ..... User directories here

- note box:

LaTeX
```
\begin{note}
This is a Note
\end{note}
```

> **Note:** This is a Note

- todo box:

LaTeX
```
\begin{note}
This is a to do statement (temporary)
\end{note}
```

> **Note:** This is a to do statement (temporary)

Chapter 9

Required input header keywords

The following FITS descriptors of the 2D raw frames are required for the DRS. Last updated version 21 Nov 2014.

9.1 Standard FITS Keywords

BITPIX	=	16	/	16bit
NAXIS	=	2	/	Number of axes
NAXIS1	=	4096	/	Number of pixel columns
NAXIS2	=	4096	/	Number of pixel rows
BZERO	=	32768.0	/	Zero factor
BSCALE	=	1.0	/	Scale factor
DATE	=	'2013-11-26T09:06:14'	/	UTC Date of file creation
INSTRUME	=	'SPIROU'	/	Instrument Name

9.2 FITS keywords related to the detector

EXPTIME	=	800.0	/	Integration time (seconds)
DARKTIME	=	800.0	/	Dark current time (seconds)
GAIN	=	1.30	/	gain (electrons/ADU)
RDNOISE	=	4.20	/	read noise (electrons)
NSUBEXP	=	4	/	Total number of sub-exposures of 5.2s
OBSTYPE	=	'NORMAL'	/	Exposure type (DARK/NORMAL)
MIDEXPTM	=	400	/	mid-exposure time (seconds)
EMCNTS	=	444578	/	exposure meter counts at end

9.3 FITS keywords related to the target

OBJNAME	=	Gl9999	/	Target name
OBJRA	=	'5:35:09.87'	/	Target right ascension
OBJDEC	=	'-5:27:53.3'	/	Target declination
OBJRAPM	=	0.560	/	Target right ascension proper motion in as/yr
OBJDECPM	=	-0.33	/	Target declination proper motion in as/yr
OBJEQUIN	=	2000.0	/	Target equinox
OBJRV	=	-30.0	/	Target Radial velocity (km/s) (999 if unknown)
OBJTYPE	=	'M5'	/	Target spectral type
OBJJMAG	=	8.2	/	Target J magnitude
OBJHMAG	=	9.2	/	Target H magnitude
OBJKMAG	=	10.0	/	Target K magnitude

9.4 FITS keywords related to the telescope

ACQTIME	=	2013-11-26T09 :06 :14.858	/	Date at start of observation
ACQTIME1	=	1385456774	/	Date in unix time at start of observation
DATE_OBS	=	'2013-11-26T09 :06 :14.858'	/	Date at start of observation (UTC)
EQUINOX	=	2000.0	/	Equinox of coordinates
EPOCH	=	2000.0	/	Epoch of coordinates
MJDATE	=	56622.3700212	/	Modified Julian Date at start of observation
MJEND	=	56622.3797593	/	Modified Julian Date at end of observation
AIRMASS	=	1.4	/	Airmass at start of observation
RA	=	'5:35:09.87'	/	Telescope right ascension
DEC	=	'-5:27:53.3'	/	Telescope declination
SEEING	=	1.0	/	Seeing at start of observation

9.5 FITS keywords related to the instrument

TPL_NAME	=	'SPIROU_POL_WAVE'	/	template Name
TPL_NEXP	=	1	/	# of exposure within template
TPL_EXPN	=	1	/	exposure # within template
INS_CAL	=	'WAVE'	/	Simultaneous calibration (WAVE/FP/NONE)
INS_LAMP	=	'UrAr'	/	Calibration lamp
INS_RHB1	=	90	/	SPIROU rhomb 1 position (deg)
INS_RHB2	=	180	/	SPIROU rhomb 2 position deg)

Chapter 10

Variables

To better understand the variables in the DRS we have laid out each variable in the following way:

- **Variable title (VARIABLE_NAME)**

  Description of the variable

  VARIABLE_NAME = Default Value

  Used in: The recipe used the variable is used in.
  Defined in: The place where the variable is defined.
  Called in: The code (module + function) where variable is used.
  Level: Who should be able to change this variable, levels are as follows:

  – Public: Everyone (including the user)

  – Private: Only the developer

Note: All variable from all configuration files are (and should be) loaded into the main parameter dictionary 'p' in all recipes and thus are accessed via:

Python/Ipython

```
variable = p["VARIABLE_NAME"]
```

10.1 Variable file locations

10.1.1 User modifiable variables

The variables are currently stored in two places. The first (../config/config.txt) contains constants that deal with initial set up. These were mentioned in Section 2.6 and are located in {INSTALL_DIR}/config/../config/config.txt.

The other variables modify how the DRS runs. These are located in constants_SPIROU.txt (located at {INSTALL_DIR}/config/constants_SPIROU.txt).

10.1.2 Private variables

In addition to the above (user modifiable public variable files) there are several files that will contain all constants that should not be changed by a user (i.e. static variables that are set and changed only in development). These are described below:

- **Keywords:** The keywords for header input and output are stored in SpirouDRS.spirouConfig .spirouKeywords. This contains keyword definitions in the form of a python list:

Python/Ipython

```
kw_VARIABLE = ['KEYWORD', 'Default value', 'Comment']
```

where the 'KEYWORD' is the key in the FITs REC header file, with the value and comment defined in the next positions. i.e. in a FITs REC header reader one would expect

```
KEYWORD = Default value / Comment
```

- **Constants and Pseudo-constants:** These are stored in SpirouDRS.spirouConfig.spirouConst, they range from simple objects (strings, integers, float, lists, python dictionaries etc) to more complicated 'pseudo-constants' that are constructed themselves from other constants. These are kept private (i.e. no mentioned in the user manual) as they should not need be changed by the average user.

10.2 Global variables

- **DRS Name (DRS_NAME)**

  Defines the data reduction software name. Value must be a valid string.

  DRS_NAME = SPIROU

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.NAME()
  Called in: All Recipes
  Level: Private

- **DRS Version (DRS_VERSION)**

  Defines the data reduction software version. Value must be a valid string.

  DRS_VERSION = 0.0.1

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.VERSION()
  Called in: All Recipes
  Level: Private

- **Package name (package)**

  Defines the name of the python package that all sub-modules are located in. Value must be a string and be the name of a valid python package.

  package = SpirouDRS

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.PACKAGE()
  Called in: All Recipes
  Level: Private

- **authors (authors)**

  Defines the authors of the DRS. Value must be a string, author names separated by a comma.

  authors = N. Cook, F. Bouchy, E. Artigau, I. Boisse, M. Hobson, C. Moutou

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.AUTHORS()
  Called in: All Recipes
  Level: Private

- **date (date)**

  Defines the last edited date for the DRS. Value must be a string in format YYYY-MM-DD format.

  date = 2017-11-17

  Used in: None
  Defined in: SpirouDRS.spirouConfig.spirouConst.LATEST_EDIT()
  Called in: None
  Level: Private

- **Plotting switch (DRS_PLOT)**

  Defines whether to show plots (A value of 1 to show plots, a value of 0 to not show plots). Value must be an integer (0 or 1) or boolean (True or False)

  DRS_PLOT = 1

  Used in: All Recipes
  Defined in: ../config/config.txt
  Called in: All Recipes
  Level: Public

- **Use matplotlib interactive plot environment (interactive_plots)**

  Defines whether to use the matplotlib interactive plot environment. If True or 1 uses 'plot.ion()' and plots do not interrupt the running of code. If False or 0 all plots are run and 'plt.show(), plt.close()' is used after each plot (pausing the code and destroying the plots after they are manually closed). This is mostly useful for debugging.

  interactive_plots = True

  Used in: SpirouDRS.spirouCore.spirouPlot
  Defined in: SpirouDRS.spirouConfig.spirouConst.INTERACITVE_PLOTS_ENABLED()
  Called in: SpirouDRS.spirouCore.spirouPlotvariable definition
  Level: Private

- **Debug mode (ic_debug)**

  Enable various numeric debug codes (0 for no debug). Value must be an integer where:

  - 0 = No debug
  - 1 = Level 1 debug

    > **TODO:** Define level 1 debug

  - 2 = Level 2 debug

    > **TODO:** Define level 2 debug

  ic_debug = 0

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: ???
  Level: Public

  > **Note:** Should this be public?

- **Plot interval (ic_display_timeout)**

  Set the interval between plots in seconds (for certain interactive graphs). Value must be a valid float larger than zero.

  ic_display_timeout = 0.5

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in:
  Level: Public

  > **Note:** Should this be public?

10.3 Directory variables

- **The data directory (TDATA)**

  Defines the path to the data directory. Value must be a string containing a valid file location.

  TDATA = /drs/data/

  Used in: All Recipes
  Defined in: ../config/config.txt
  Called in: SpirouDRS.spirouConfig.spirouConst
  Level: Public

- **The installation directory (DRS_ROOT)**

  Defines the installation directory ({INSTALL_DIR}). Value must be a string containing a valid file location.

  DRS_ROOT = /drs/INTROOT/

  Used in: All Recipes
  Defined in: ../config/config.txt
  Called in: SpirouDRS.spirouConfig.spirouConst
  Level: Public

- **The raw data directory (DRS_DATA_RAW)**

  Defines the directory that the reduced data will be saved to/read from. Value must be a string containing a valid file location.

  DRS_DATA_RAW = /drs/data/raw

  Used in: All Recipes
  Defined in: ../config/config.txt
  Called in: SpirouDRS.spirouConfig.spirouConst
  Level: Public

- **The reduced data directory (DRS_DATA_REDUC)**

  Defines the directory that the reduced data will be saved to/read from. Value must be a string containing a valid file location.

  DRS_DATA_REDUC = /drs/data/reduced

  Used in: All Recipes
  Defined in: ../config/config.txt
  Called in: SpirouDRS.spirouConfig.spirouConst
  Level: Public

- **The calibration database and calibration file directory (DRS_CALIB_DB)**

  Defines the directory that the calibration files and database will be saved to/read from. Value must be a string containing a valid file location.

  DRS_CALIB_DB = /drs/data/calibDB

  Used in: All Recipes
  Defined in: ../config/config.txt
  Called in: SpirouDRS.spirouConfig.spirouConst
  Level: Public

- **The log directory (DRS_DATA_MSG)**

  Defines the directory that the log messages are stored in. Value must be a string containing a valid file location.

  DRS_DATA_MSG = /drs/data/msg

  Used in: All Recipes
  Defined in: ../config/config.txt
  Called in: SpirouDRS.spirouConfig.spirouConst
  Level: Public

- **The working directory (DRS_DATA_WORKING)**

  Defines the working directory. Value must be a string containing a valid file location.

  DRS_DATA_WORKING = /drs/data/tmp/

  Used in: All Recipes
  Defined in: ../config/config.txt
  Called in: SpirouDRS.spirouConfig.spirouConst
  Level: Public

10.4 Image variables

- **Resizing blue window (ic_ccd{x/y}_blue_{low/high})**

  The blue window used in cal_DARK_spirou. Each value must be a integer between 0 and the maximum array size in each dimension.

  ic_ccdx_blue_low = 2048-200
  ic_ccdx_blue_high = 2048-1500
  ic_ccdy_blue_low = 2048-20
  ic_ccdy_blue_high = 2048-350

  Used in: cal_DARK_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DARK_spirou.main()
  Level: Public

- **Resizing red window (ic_ccd{x/y}_red_{low/high})**

  The blue window used in cal_DARK_spirou. Each value must be a integer between 0 and the maximum array size in each dimension.

  ic_ccdx_red_low = 2048-20
  ic_ccdx_red_high = 2048-1750
  ic_ccdy_red_low = 2048-1570
  ic_ccdy_red_high = 2048-1910

  Used in: cal_DARK_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DARK_spirou.main()
  Level: Public

- **Resizing red window (ic_ccd{x/y}_{low/high})**

  The blue window used in cal_DARK_spirou. Each value must be a integer between 0 and the maximum array size in each dimension.

  ic_ccdx_low = 5
  ic_ccdx_high = 2040
  ic_ccdy_low = 5
  ic_ccdy_high = 1935

  Used in: cal_loc_RAW_spirou, cal_SLIT_spirou, cal_FF_RAW_spirou, cal_extract_RAW_spirou, cal_DRIFT_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spiroumain(), cal_SLIT_spiroumain(), cal_FF_RAW_spiroumain(), cal_extract_RAW_spiroumain(), cal_DRIFT_RAW_spirou.main()
  Level: Public

- **Available fiber types (fiber_types)**

  Defines the type of fiber we have (used in various codes). Theses are define in a python list of string, where the earlier a fiber is in the list the more it takes priority in searches (i.e. AB over A or B if AB is first)

  fiber_types = `['AB', 'A', 'B', 'C']`

  Used in: cal_extract_RAW_spirou, cal_DRIFT_E2DS_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_extract_RAW_spirou.main(), SpirouDRS.spirouStartup.spirouStartup.get_fiber_type()
  Level: Public

10.5 Fiber variables

These variables are defined for each type of fiber and thus are defined as a python dictionary of values (read using the python 'eval' function) . As such they all must contain the same dictionary keys (currently 'AB', 'A', 'B' and 'C').

> **Note:** For python to combine these at run time the suffix '_fpall' must be used (thus once a fiber is defined the code will know to extract the key before the suffix). i.e. for variable 'nbfib_fpall' and a fiber 'AB' the extracted parameter will be 'nbfib' with the value in the dictionary corresponding to the 'AB' key.

- **Number of fibers (nbfib_fpall)**

  This describes the number of fibers of a given type. Must be a python dictionary with identical keys to all other fiber parameters (each value must be an integer).

  nbfib_fpall = `{'AB':2, 'A':1, 'B':1, 'C':1}`

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.main()
  Level: Public

- **Order skip number (ic_first_order_jump_fpall)**

  Describes the number of orders to skip at the start of an image. Must be a python dictionary with identical keys to all other fiber parameters (each value must be an integer).

  ic_first_order_jump_fpall = `{'AB':2, 'A':0, 'B':0, 'C':0}`

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.main()
  Level: Public

- **Maximum order numbers (ic_locnbmaxo_fpall)**

  Describes the maximum allowed number of orders. Must be a python dictionary with identical keys to all other fiber parameters (each value must be an integer).

  ic_locnbmaxo_fpall = `{'AB':72, 'A':36, 'B':36, 'C':36}`

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.main()
  Level: Public

- **Number of orders to fit (QC) (qc_loc_nbo_fpall)**

  Quality control parameter for the number of orders on fiber to fit. Must be a python dictionary with identical keys to all other fiber parameters (each value must be an integer).

  qc_loc_nbo_fpall = `{'AB':72, 'A':36, 'B':36, 'C':36}`

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.main()
  Level: Public

  Note: Should this be merged with 'ic_locnbmaxo_fpall'?

- **Fiber types for this fiber (fib_type_fpall)**

  The fiber type(s) – as a list – for this fiber. Must be a python dictionary with identical keys to all other fiber parameters (each value must be a list of strings).

  fib_type_fpall = `{'AB':["AB"], 'A':["A"], 'B':["B"], 'C':["C"]}`

  Used in: cal_FF_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_FF_RAW_spirou.main()
  Level: Public

  Note: This is not be needed but is in here due to a loop in cal_FF_RAW_spirou

- **Half-zone extraction width (left/top) (ic_ext_range1_fpall)**

  The pixels are extracted from the center of the order out to the edges in the row direction (y-axis), i.e. defines the illuminated part of the order - this number defines the **top** side (if one requires a symmetric extraction around the order fit both range 1 and range 2 – below – should be the same). This can also be used to extract A and B separately (where the fit order is defined at the center of the AB pair). Must be a python dictionary with identical keys to all other fiber parameters.

  ic_ext_range1_fpall = `{'AB':14.5, 'A':0.0, 'B':14.5, 'C':7.5}`

  Used in: cal_FF_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_extract_RAW_spirou.main(), SpirouDRS.spirouEXTOR.spirouEXTOR.extract_tilt_weight_order2(), SpirouDRS.spirouCore.spirouPlot.ff_sorder_fit_edges()
  Level: Public

  Note: Formally this was called 'plage1' in cal_FF_RAW_spirou

- **Half-zone extraction width (right/bottom) (ic_ext_range2_fpall)**

The pixels are extracted from the center of the order out to the edges in the row direction (y-axis), i.e. defines the illuminated part of the order - this number defines the **bottom** side (if one requires a symmetric extraction around the order fit both range 1 and range 2 – below – should be the same). This can also be used to extract A and B separately (where the fit order is defined at the center of the AB pair). Must be a python dictionary with identical keys to all other fiber parameters.

ic_ext_range2_fpall = `{'AB':14.5, 'A':14.5, 'B':0.0, 'C':7.5}`

Used in:	cal_FF_RAW_spirou, cal_extract_RAW_spirou
Defined in:	constants_SPIROU.txt
Called in:	cal_FF_RAW_spirou.main(), cal_extract_RAW_spirou.main(), SpirouDRS.spirouEXTOR.spirouEXTOR.extract_tilt_weight_order2(), SpirouDRS.spirouCore.spirouPlot.ff_sorder_fit_edges()
Level:	Public

Note: Formally this was called 'plage2' in cal_FF_RAW_spirou

- **Half-zone extraction width for full extraction (ic_ext_range_fpall)**

The pixels are extracted from the center of the order out to the edges in the row direction (y-axis), i.e. defines the illuminated part of the order. In cal_extract_RAW_spirouboth sides of the fit order are extracted at with the same width (symmetric). Must be a python dictionary with identical keys to all other fiber parameters.

ic_ext_range_fpall = `{'AB':14.5, 'A':14.5, 'B':14.5, 'C':7.5}`

Used in:	cal_extract_RAW_spirou
Defined in:	constants_SPIROU.txt
Called in:	SpirouDRS.spirouEXTOR.spirouEXTOR.extract_order(), SpirouDRS.spirouEXTOR.spirouEXTOR.extract_tilt_order(), SpirouDRS.spirouEXTOR.spirouEXTOR.extract_tilt_weight_order(), SpirouDRS.spirouEXTOR.spirouEXTOR.extract_weight_order()
Level:	Public

Note: Formally this was called 'plage' in cal_extract_RAW_spirou

- **Localization fiber for extraction (loc_file_fpall)**

  Defines the localization fiber to use for each fiber type. This is the file in calibDB that is used i.e. the keyword master_calib_SPIROU.txt used will be ÒLOC_{loc_file_fpall}' (e.g. for fiber='AB' use 'LOC_AB'). Must be a python dictionary with identical keys to all other fiber parameters.

  loc_file_fpall = `{'AB':'AB', 'A':'AB', 'B':'AB', 'C':'C'}`

  Used in: cal_extract_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouLOCOR.spirouLOCOR.get_loc_coefficients()
  Level: Public

- **Order profile fiber for extraction (orderp_file_fpall)**

  Defines the order profile fiber to use for each fiber type. This is the file in calibDB that is used i.e. the keyword master_calib_SPIROU.txt used will be ÒRDER_PROFILE_{orderp_file_fpall}' (e.g. for fiber='AB' use 'ORDER_PROFILE_AB'). Must be a python dictionary with identical keys to all other fiber parameters.

  orderp_file_fpall = `{'AB':'AB', 'A':'AB', 'B':'AB', 'C':'C'}`

  Used in: cal_extract_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouImage.spirouFITS.read_order_profile_superposition()
  Level: Public

- **Half-zone extract width cal_DRIFT_RAW_spirou (ic_ext_d_range_fpall)**

  The size in pixels of the extraction away from the order localization fit (to the top and bottom) - defines the illuminated area of the order for extraction. Must be a python dictionary with identical keys to all other fiber parameters.

  ic_ext_d_range_fpall = `{'AB':14.0, 'A':14.0, 'B':14.0, 'C':7.0}`

  Used in: cal_DRIFT_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DRIFT_RAW_spirou.main()
  Level: Public

  Note: Formally this was called 'ic_extnbsig' in cal_DRIFT_RAW_spirou

10.6 Dark calibration variables

- **Lower percentile for dead pixel stats (dark_qmin)**

  This defines the lower percentile to be logged for the fraction of dead pixels statistics. Value must be an integer between 0 and 100 (1 sigma below the mean is ~16).

  dark_qmin = 5

  Used in: cal_DARK_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouImage.spirouImage.measure_dark()
  Level: Public

- **Upper percentile for dead pixel stats (dark_qmax)**

  This defines the upper percentile to be logged for the fraction of dead pixels statistics. Value must be an integer between 0 and 100 (1 sigma above the mean is ~84).

  dark_qmax = 95

  Used in: cal_DARK_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouImage.spirouImage.measure_dark()
  Level: Public

- **Dark stat histogram bins (histo_bins)**

  Defines the number of bins to use in the dark histogram plot. Value must be a positive integer.

  histo_bins = 200

  Used in: cal_DARK_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouImage.spirouImage.measure_dark()
  Level: Public

- **Lower bound for the Dark stat histogram (histo_range_low)**

  Defines the lower bound for the dark statistic histogram. Value must be a float less than (no equal to) the value of 'histo_range_high'

  histo_range_low = -0.5

  Used in: cal_DARK_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouImage.spirouImage.measure_dark()
  Level: Public

- **Upper bound for the Dark stat histogram (histo_range_high)**

  Defines the upper bound for the dark statistic histogram. Value must be a float greater than (not equal to) the value of 'histo_range_low'

  histo_range_high = 5

  Used in: cal_DARK_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouImage.spirouImage.measure_dark()
  Level: Public

- **Bad pixel cut limit (dark_cutlimit)**

  Defines the bad pixel cut limit in ADU/s.

  $$badpixels = (image > \text{dark\_cut\_limit}) \text{ OR (non-finite)} \quad (10.1)$$

  dark_cutlimit = 100.0

  Used in: cal_DARK_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DARK_spirou.main()
  Level: Public

10.7 Localization calibration variables

- **Order profile smoothed box size (loc_box_size)**

  Defines the size of the order profile smoothing box (from the central pixel minus size to the central pixel plus size). Value must be an integer larger than zero.

  loc_box_size = 10

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.main()
  Level: Public

- **Image row offset (ic_offset)**

  The row number (y axis) of the image to start localization at (below this row orders will not be fit). Value must be an integer equal to or larger than zero.

  ic_offset = 40

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.main()
  Level: Public

- **Central column of the image (ic_cent_col)**

  The column which is to be used as the central column (x-axis), this is the column that is initially used to find the order locations. Value must be an integer between 0 and the number of columns (x-axis dimension).

  ic_cent_col = 1000

  Used in: cal_loc_RAW_spirou, cal_FF_RAW_spirou, cal_extract_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.main(), cal_FF_RAW_spirou.main(), cal_extract_RAW_spirou.main(), SpirouDRS.spirouBACK.spirouBACK.measure_background_and_get_central_pixels(), SpirouDRS.spirouCore.spirouPlot.slit_sorder_plot(), SpirouDRS.spirouEXTOR.spirouEXTOR.extract_AB_order(), SpirouDRS.spirouLOCOR.spirouLOCOR.find_order_centers(), SpirouDRS.spirouLOCOR.spirouLOCOR.initial_order_fit()
  Level: Public

- **Localization window row size (ic_ext_window)**

Defines the size of the localization window in rows (y-axis). Value must be an integer larger than zero and less than the number of rows (y-axis dimension).

ic_ext_window = 12

Used in: cal_loc_RAW_spirou
Defined in: constants_SPIROU.txt
Called in: SpirouDRS.spirouLOCOR.spirouLOCOR.find_order_centers
Level: Public

Note: Formally this was called 'ic_ccdcolc' in cal_loc_RAW_spirou

- **Localization window column step (ic_locstepc)**

For the initial localization procedure interval points along the order (x-axis) are defined and the centers are found, this is used as the first estimate of the order shape. This parameter defines that interval step in columns (x-axis). Value must be an integer larger than zero and less than the number of columns (x-axis dimension).

ic_locstepc = 12

Used in: cal_loc_RAW_spirou
Defined in: constants_SPIROU.txt
Called in: SpirouDRS.spirouLOCOR.spirouLOCOR.find_order_centers
Level: Public

- **Image gap index (ic_image_gap)**

Defines the image gap index. The order is skipped if the top of the row (row number - ic_ext_window) or bottom of the row (row number + ic_ext_window) is inside this image gap index. i.e. a order is skipped if:

$$\text{(top of the row} < \text{ic\_image\_gap) OR (bottom of the row} > \text{ic\_image\_gap)} \tag{10.2}$$

Value must be an integer between zero and the number of rows (y-axis dimension).

ic_image_gap = 0

Used in: cal_loc_RAW_spirou
Defined in: constants_SPIROU.txt
Called in: SpirouDRS.spirouLOCOR.spirouLOCOR.find_order_centers
Level: Public

Note: This is set to zero and never used in a meaningful way, should it be removed?

- **Minimum order row size (ic_widthmin)**

  Defines the minimum row width (width in y-axis) to accept an order as valid. If below this threshold order is not recorded. Value must be an integer between zero and the number of rows (y-axis dimension).

  ic_widthmin = 5

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouLOCOR.spirouLOCOR.find_order_centers
  Level: Public

- **Min/Max smoothing box size (ic_locnbpix)**

  Defines the half-size of the rows to use when smoothing the image to work out the minimum and maximum pixel values. This defines the half-spacing between orders and is used to estimate background and the maximum signal. Value must be greater than zero and less than the number of rows (y-axis dimension).

  ic_locnbpix = 45

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouBACK.spirouBACK.measure_min_max()
  Level: Public

- **Minimum signal amplitude (ic_min_amplitude)**

  Defines a cut off (in e-) where below this point the central pixel values will be set to zero. Value must be a float greater than zero.

  ic_min_amplitude = 100.0

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouBACK.spirouBACK.measure_background_and_get_central_pixels()
  Level: Public

- **Normalized background amplitude threshold (ic_locseuil)**

Defines the normalized amplitude threshold to accept pixels for background calculation (pixels below this normalized value will be used for the background calculation). Value must be a float between zero and one.

ic_locseuil = 0.2

Used in: cal_loc_RAW_spirou
Defined in: constants_SPIROU.txt
Called in: SpirouDRS.spirouBACK.spirouBACK.measure_background_and_get_central_pixels()
Level: Public

- **Saturation threshold on the order profile plot (ic_satseuil)**

Defines the saturation threshold on the order profile plot, pixels above this value will be set this value (ic_satseuil). Value must be a float greater than zero.

ic_satseuil = 64536

Used in: cal_loc_RAW_spirou
Defined in: constants_SPIROU.txt
Called in: cal_loc_RAW_spirou.main()
Level: Public

- **Degree of the fitting polynomial for localization position (ic_locdfitc)**

Defines the degree of the fitting polynomial for locating the positions of each order i.e. if value is 1 is a linear fit, if the value is 2 is a quadratic fit. The value must be a positive integer equal to or greater than zero (zero would lead to a constant fit along the column direction (x-axis direction).

ic_locdfitc = 5

Used in: cal_loc_RAW_spirou
Defined in: constants_SPIROU.txt
Called in: SpirouDRS.spirouLOCOR.spirouLOCOR.initial_order_fit(), SpirouDRS.spirouLOCOR.spirouLOCOR.sigmaclip_order_fit()
Level: Public

- **Degree of the fitting polynomial for localization width (ic_locdfitw)**

  Defines the degree of the fitting polynomial for measuring the width of each order i.e. if value is 1 is a linear fit, if the value is 2 is a quadratic fit. The value must be a positive integer equal to or greater than zero (zero would lead to a constant fit along the row direction (y-axis direction).

  ic_locdfitw = 5

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouLOCOR.spirouLOCOR.initial_order_fit(), SpirouDRS.spirouLOCOR.spirouLOCOR.sigmaclip_order_fit()
  Level: Public

- **Degree of the fitting polynomial for localization position error (ic_locdfitp)**

  Defines the degree of the fitting polynomial for locating the positions error of each order i.e. if value is 1 is a linear fit, if the value is 2 is a quadratic fit. The value must be a positive integer equal to or greater than zero (zero would lead to a constant fit along the column direction (x-axis direction).

  ic_locdfitp = 3

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouConfig.spirouKeywords, cal_loc_RAW_spirou.main(), SpirouDRS.spirouLOCOR.spirouLOCOR.sigmaclip_order_fit()
  Level: Public

  Note: This is only currently used to add the value to the localization file ('_loco_{fiber}.fits') but not used in any calculation. It could be removed?

- **Maximum RMS for sigma-clipping order fit (positions) (ic_max_rms_center)**

  Defines the maximum RMS allowed for an order, if RMS is above this value the position with the highest residual is removed and the fit is recalculated without that position (sigma-clipped). Value must be a positive float. i.e. position fit is recalculated if:

  $$max(RMS) > \text{ic\_max\_rms\_center} \tag{10.3}$$

  ic_max_rms_center = 0.2

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouLOCOR.spirouLOCOR.sigmaclip_order_fit()
  Level: Public

- **Maximum peak-to-peak for sigma-clipping order fit (positions) (ic_max_ptp_center)**

  Defines the maximum peak-to-peak value allowed for an order, if the peak to peak is above this value the position with the highest residual is removed and the fit is recalculated without that position (sigma-clipped). Value must be a positive float. i.e. position fit is recalculated if:

  $$max(|\text{residuals}|) > \text{ic\_max\_ptp\_center} \tag{10.4}$$

  ic_max_ptp_center = 0.2

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouLOCOR.spirouLOCOR.sigmaclip_order_fit()
  Level: Public

- **Maximum peak-to-peak-RMS ratio for sigma-clipping order fit(positions) (ic_ptporms_center)**

  Defines the maximum ratio of peak-to-peak residuals and rms value allowed for an order, if the ratio is above this value the position with the highest residual is removed and the fit is recalculated without that position (sigma-clipped). Value must be a positive float. i.e. position fit is recalculated if:

  $$max(|\text{residuals}|)/\text{RMS} > \text{ic\_ptporms\_center} \tag{10.5}$$

  ic_ptporms_center = 8.0

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouLOCOR.spirouLOCOR.sigmaclip_order_fit()
  Level: Public

- **Maximum RMS for sigma-clipping order fit (width) (ic_max_rms_fwhm)**

  Defines the maximum RMS allowed for an order, if RMS is above this value the width with the highest residual is removed and the fit is recalculated without that width (sigma-clipped). Value must be a positive float. i.e. width fit is recalculated if:

  $$max(RMS) > \text{ic\_max\_rms\_width} \tag{10.6}$$

  ic_max_rms_fwhm = 1.0

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouLOCOR.spirouLOCOR.sigmaclip_order_fit()
  Level: Public

- **Maximum peak-to-peak for sigma-clipping order fit (widths) (ic_max_ptp_fracfwhm)**

  Defines the maximum peak-to-peak value allowed for an order, if the peak to peak is above this value the width with the highest residual is removed and the fit is recalculated without that width (sigma-clipped). Value must be a positive float. i.e. width fit is recalculated if:

  $$max(|\text{residuals/data}|) \times 100 > \text{ic\_max\_ptp\_fracfwhm} \tag{10.7}$$

  ic_max_ptp_fracfwhm = 1.0

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouLOCOR.spirouLOCOR.sigmaclip_order_fit()
  Level: Public

- **Delta width 3 convolve shape model (ic_loc_delta_width)**

  Defines the delta width in pixels for the 3 convolve shape model - currently not used. Value must be a positive float.

  ic_loc_delta_width = 1.85

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.main(), SpirouDRS.spirouConfig.spirouKeywords
  Level: Public

  Note: This is currently not used (other than saving in the calibDB loco file. Can it be removed?).

- **Localization archiving option (ic_locopt1)**

  Whether we save the location image with the superposition of the fit (zeros). If this option is 1 or True it will save the file to '_with-order_{fiber}.fits' if 0 or False it will not save this file. Value must be 1, 0, True or False.

  ic_locopt1 = 1

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.main()
  Level: Public

10.8 Slit calibration variables

- **Tilt oversampling factor (ic_tilt_coi)**

  Defines the oversampling factor used to work out the tilt of the slit. Value must be an integer value larger than zero.

  ic_tilt_coi = 10

  Used in: cal_SLIT_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouImage.spirouImage.get_tilt()
  Level: Public

  Note: Formally this was called 'coi' in cal_SLIT_spirou.

- **Slit fit order plot offset factor (ic_facdec)**

  Defines an offset of the position fit to show the edges of the illuminated area. (Final offset is ±× 2 of this offset away from the order fit. Value must be a positive float.)

  ic_facdec = 1.6

  Used in: cal_SLIT_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouCore.spirouPlot.slit_sorder_plot()
  Level: Public

- **Degree of the fitting polynomial for the tilt (ic_tilt_fit)**

  Defines the degree of the fitting polynomial for determining the tilt i.e. i.e. if value is 1 is a linear fit, if the value is 2 is a quadratic fit. The value must be a positive integer equal to or greater than zero (zero would lead to a constant fit).

  ic_tilt_fit = 3

  Used in: cal_SLIT_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouImage.spirouImage.fit_tilt()
  Level: Public

- **Selected order in Slit fit order plot (ic_slit_order_plot)**

  Defines the selected order to plot the fit for in the Slit fir order plot. Value must be between zero and the maximum number of orders.

  ic_slit_order_plot = 10

  Used in: cal_SLIT_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouCore.spirouPlot.slit_sorder_plot()
  Level: Public

10.9 Flat fielding calibration variables

- **Measure background (ic_do_bkgr_subtraction)**

  Define whether to measure the background and do a background subtraction. Value must be True or 1 to do the background measurement and subtraction or be False or 0 to not do the background measurement and subtraction.

  ic_do_bkgr_subtraction = 0

  Used in: cal_FF_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_FF_RAW_spirou.main()
  Level: Public

  Note: Currently even if True or 1 the background is not calculated as the interpol function has not been converted to python.

- **Half-size of background window (ic_bkgr_window)**

  Defines the half-size (in pixels) of the background window to create a sub-frame to find the minimum 2× ic_bkgr_window pixels for which to calculate the background from. Size is used in both row and column (y and x) direction. Value must be an integer between zero and the minimum(row number, column number) (minimum(x-axis dimension, y-axis dimension)).

  ic_bkgr_window = 100

  Used in: cal_FF_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouBACK.spirouBACK.measure_background_flatfield()
  Level: Public

- **Number of orders in tilt measurement (ic_tilt_nbo)**

  Defines the number of orders in the tilt measurement file (TILT key in the master_calib_SPIROU.txt). This is the number of tilts that will be extracted. Value must be an integer larger than zero and smaller than or equal to the total number of orders present in the TILT file.

  ic_tilt_nbo = 36

  Used in: cal_FF_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouImage.spirouFITS.read_tilt_file()
  Level: Public

  Note: This can probably be removed and replaced with a check to the TILT file - to automatically determine how many orders there should be.

Note: This was formally called 'nbo' and was hard coded in cal_FF_RAW_spirou.

- **The manually set sigdet for flat fielding. (ic_ff_sigdet)**

  This defines the sigdet to use in the weighted tilt extraction. Set to -1 to use from the input file ('fitsfilename') HEADER. Value must be either -1 or a positive float.

  ic_ff_sigdet = 100.0

  Used in: cal_FF_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_FF_RAW_spirou.main()
  Level: Public

- **Half size blaze window (ic_extfblaz)**

  Defines the distance from the central column that should be used to measure the blaze for each order. Value must be an integer greater than zero and less than half the number of columns (x-axis dimension).

  ic_extfblaz = 50

  Used in: cal_FF_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_FF_RAW_spirou.main()
  Level: Public

- **Fit degree for the blaze polynomial fit (ic_blaze_fitn)**

  Defines the degree of the fitting polynomial for fitting the blaze function of each order i.e. if value is 1 is a linear fit, if the value is 2 is a quadratic fit. The value must be a positive integer equal to or greater than zero (zero would lead to a constant fit along the column direction (x-axis direction).

  ic_blaze_fitn = 5

  Used in: cal_FF_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_FF_RAW_spirou.main()
  Level: Public

- **Selected order for flat fielding plot (ic_ff_order_plot)**

Defines the selected order to plot on the flat fielding image plot. Value must be a integer between zero and the number of orders.

ic_ff_order_plot = 5

Used in: cal_FF_RAW_spirou
Defined in: constants_SPIROU.txt
Called in: SpirouDRS.spirouCore.spirouPlot.ff_sorder_fit_edges
Level: Public

Note: This was formally called 'ic_plot_order' in cal_FF_RAW_spirou.

- **Plot all order fits for flat fielding plot (ic_ff_plot_all_orders)**

If True or 1, instead of plotting the selected order from ic_ff_order_plot will plot the order fits (and edges) for all orders. This is slower than just plotting one. Value must be True or 1 or False or 0.

ic_ff_plot_all_orders = 0

Used in: cal_FF_RAW_spirou
Defined in: constants_SPIROU.txt
Called in: cal_FF_RAW_spirou.main()
Level: Public

Note: This is a new plot, instead of plotting one selected order plots all orders - this is obviously slightly slower than just plotting one example order.

10.10 Extraction calibration variables

- **Extraction option - rough extraction (ic_extopt)**

  Extraction option for rough extraction:

  - if 0 extraction by summation over a constant range
  - if 1 extraction by summation over constants sigma (not currently available)
  - if 2 horne extraction without cosmic elimination (not currently available)
  - if 3 horne extraction with cosmic elimination (not currently available)

  Used for estimating the slit tilt and in calculating the blaze/flat fielding. Value must be a integer between 0 and 3.

  ic_extopt = 0

  Used in: cal_SLIT_spirou, cal_FF_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouEXTOR.spirouEXTOR.extract_AB_order(), SpirouDRS.spirouEXTOR.spirouEXTOR.extract_order
  Level: Public

- **Extraction distance - rough extraction (ic_extnbsig)**

  The pixels are extracted from the center of the order out to the edges in the row direction (y-axis), i.e. defines the illuminated part of the order). Used for estimating the slit tilt and in calculating the blaze/flat fielding. Value must be a positive float between 0 and the total number of rows (y-axis dimension).

  ic_extnbsig = 2.5

  Used in: cal_SLIT_spirou, cal_FF_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouEXTOR.spirouEXTOR.extract_AB_order
  Level: Public

- **Extraction type (ic_extact_type)**

  Defines which type of extract should be used in cal_extract_RAW_spirou. This variable is overwritten if using cal_extract_RAW_spirouABor cal_extract_RAW_spirouC. The value must be one of the following:

  - simple ..... just does extraction as is.
  - tilt ..... does the extraction and corrects for tilt
  - weight ..... does the extraction with a weighting for bad pixels
  - tiltweight ..... does the extraction + 'tilt' + 'weight'
  - all ..... performs all extractions (saves separately). The E2DS file='weight'.

  Value should be a python string with one of the above values only. Any other value will cause an error and a recipe to exit.

  ic_extact_type = tiltweight

  Used in: cal_extract_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_extract_RAW_spirou.main()
  Level: Public

Note: For all we should probably use tiltweight but as cal_extract_RAW_spirouABand cal_extract_RAW_spirouCcurrently use weight for the E2DS this is set to reproduce this.

- **Manually set the extraction sigdet (ic_ext_sigdet)**

  Set the sigdet used in the extraction process instead of using the sigdet in the FITS rec HEADER file. If the value is set to -1 the sigdet from the HEADER is used instead.

  ic_ext_sigdet = 100

  Used in: cal_extract_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_extract_RAW_spirou.main()
  Level: Public

Note: Why is this value used and not the value in the header file?

- **Selected order in extract fit order plot (ic_ext_order_plot)**

  Defines the selected order to plot the fit for in the extract fit order plot. Value must be between zero and the maximum number of orders.

  ic_ext_order_plot = 20

  Used in: cal_extract_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouCore.spirouPlot. ext_selected_order_plot()
  Level: Public

10.11 Drift calibration variables

- **Noise value for SNR drift calculation (ic_drift_noise)**

  Define the noise value for the signal to noise ratio in the drift calculation.

  $$snr = flux/\sqrt{(\text{flux} + \text{noise}^2)} \tag{10.8}$$

  Value must be a float larger than zero.

  ic_drift_noise = 100.0

  Used in: cal_DRIFT_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DRIFT_RAW_spirou.main()
  Level: Public

- **The maximum flux for a good (unsaturated) pixel (ic_drift_maxflux)**

  Defines the maximum flux to define a good pixel. This pixels and those that surround it will not be used in determining the RV parameters. Value must be a float greater than zero.

  ic_drift_maxflux = 1.e9

  Used in: cal_DRIFT_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DRIFT_RAW_spirou.main()
  Level: Public

- **Saturated pixel flag size (ic_drift_boxsize)**

  Defines the number of pixels around a saturated pixel to flag as unusable (and hence not used in determining the RV parameters). Value must be a integer larger than zero.

  ic_drift_boxsize = 12

  Used in: cal_DRIFT_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DRIFT_RAW_spirou.main()
  Level: Public

- **Large number of files for skip (drift_nlarge)**

Defines the number of files that is large enough to require the 'drift_file_skip' parameter (only uses one file in every 'drift_file_skip' files). This is done to speed up the code and avoid a bug. Value must be an integer larger than zero.

drift_nlarge = 300

Used in: cal_DRIFT_RAW_spirou, cal_DRIFT_E2DS_spirou, cal_DRIFT-PEAK_E2DS_spirou
Defined in: constants_SPIROU.txt
Called in: cal_DRIFT_RAW_spirou.main(), cal_DRIFT_E2DS_spirou.main(), cal_DRIFT-PEAK_E2DS_spirou.main()
Level: Public

Note: Has this bug been fixed, do we need to skip for a large number of files?

- **Large number of files skip parameter (cal_DRIFT_RAW_spirou) (drift_file_skip)**

Defines how many files we skip. This is done by selecting one file every 'drift_file_skip' files. i.e. if skip is 3 the code uses every 3rd file to calculate the drift. Value must be an integer larger than zero. A value of 1 is equivalent to no skipping of files regardless of the file number.

drift_file_skip = 3

Used in: cal_DRIFT_RAW_spirou
Defined in: constants_SPIROU.txt
Called in: cal_DRIFT_RAW_spirou.main()
Level: Public

- **Large number of files skip parameter (cal_DRIFT_E2DS_spirou) (drift_e2ds_file_skip)**

Defines how many files we skip. This is done by selecting one file every 'drift_file_skip' files. i.e. if skip is 3 the code uses every 3rd file to calculate the drift. Value must be an integer larger than zero. A value of 1 is equivalent to no skipping of files regardless of the file number.

drift_e2ds_file_skip = 1

Used in: cal_DRIFT_E2DS_spirou
Defined in: constants_SPIROU.txt
Called in: cal_DRIFT_E2DS_spirou.main()
Level: Public

- **Number of sigmas to cut in cosmic renormalization (cal_DRIFT_RAW_spirou) (ic_drift_cut_raw)**

  Defines the number of standard deviations to remove fluxes at (and replace with the reference flux) for cal_DRIFT_RAW_spirou. Value must be a float larger than zero.

  ic_drift_cut_raw = 3

  Used in: cal_DRIFT_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DRIFT_RAW_spirou.main()
  Level: Public

- **Number of sigmas to cut in cosmic renormalization (cal_DRIFT_E2DS_spirou) (ic_drift_cut_e2ds)**

  Defines the number of standard deviations to remove fluxes at (and replace with the reference flux) for cal_DRIFT_E2DS_spirou. Value must be a float larger than zero.

  ic_drift_cut_e2ds = 4.5

  Used in: cal_DRIFT_E2DS_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DRIFT_E2DS_spirou.main()
  Level: Public

- **Number of orders to use in drift (ic_drift_n_order_max)**

  Defines the number of orders to use (starting from zero to maximum number). This is used to get the median drift. Value must be an integer between 0 and the maximum number of orders.

  ic_drift_n_order_max = 28

  Used in: cal_DRIFT_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DRIFT_RAW_spirou.main()
  Level: Public

- **Define the way to combine orders for drift (for cal_DRIFT_RAW_spirou) (ic_drift_type_raw)**

  Defines the way to calculate the combine order drifts (to one drift per image) should either be 'weighted mean' (Equation 10.9) or 'median' (Equation 10.10) for cal_DRIFT_RAW_spirou.

  $$\text{drift} = \frac{\sum(\text{drift}_i * w_i)}{\sum w_i} \tag{10.9}$$

  where w_i is $1/\Delta v_{rms}$

  $$\text{drift} = \text{median}(\text{drift}_i) \tag{10.10}$$

  Value should be a valid python string either 'median' or 'weighted mean'.

  ic_drift_type_raw = median

Used in:	cal_DRIFT_RAW_spirou
Defined in:	constants_SPIROU.txt
Called in:	cal_DRIFT_RAW_spirou.main()
Level:	Public

- **Define the way to combine orders for drift cal_DRIFT_E2DS_spirou) (ic_drift_type_e2ds)**

  Defines the way to calculate the combine order drifts (to one drift per image) should either be 'weighted mean' (Equation 10.11) or 'median' (Equation 10.12) for cal_DRIFT_E2DS_spirou.

  $$\text{drift} = \frac{\sum(\text{drift}_i * w_i)}{\sum w_i} \tag{10.11}$$

  where w_i is $1/\Delta v_{rms}$

  $$\text{drift} = \text{median}(\text{drift}_i) \tag{10.12}$$

  Value should be a valid python string either 'median' or 'weighted mean'.

  ic_drift_type_e2ds = weighted mean

Used in:	cal_DRIFT_E2DS_spirou
Defined in:	constants_SPIROU.txt
Called in:	cal_DRIFT_E2DS_spirou.main()
Level:	Public

- **Selected order in drift fit order plot (ic_drift_order_plot)**

  Defines the selected order to plot the fit for in the drift fit order plot. Value must be between zero and the maximum number of orders.

  ic_drift_order_plot = 20

Used in:	cal_DRIFT_RAW_spirou, cal_DRIFT_E2DS_spirou
Defined in:	constants_SPIROU.txt
Called in:	SpirouDRS.spirouCore.spirouPlot. drift_plot_selected_wave_ref()
Level:	Public

10.12 Drift-Peak calibration variables

- **First order to use in drift-peak (ic_drift_peak_n_order_min)**

  Defines the first order to use (from this to {ic_drift_peak_n_order_max}). This is used to get the median drift. Value must be an integer greater than or equal to 0 and less than {ic_drift_peak_n_order_max}.

  ic_drift_peak_n_order_min = 2

  Used in: cal_DRIFT-PEAK_E2DS_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DRIFT-PEAK_E2DS_spirou.main()
  Level: Public

- **Last order to use in drift-peak (ic_drift_peak_n_order_max)**

  Defines the last order to use (from {ic_drift_peak_n_order_min} to this). This is used to get the median drift. Value must be an integer greater than {ic_drift_peak_n_order_min} and less than or equal to the maximum number of orders.

  ic_drift_peak_n_order_max = 30

  Used in: cal_DRIFT-PEAK_E2DS_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DRIFT-PEAK_E2DS_spirou.main()
  Level: Public

- **Large number of files skip parameter (cal_DRIFT_E2DS_spirou) (drift_e2ds_file_skip)**

  Defines how many files we skip. This is done by selecting one file every 'drift_file_skip' files. i.e. if skip is 3 the code uses every 3rd file to calculate the drift. Value must be an integer larger than zero. A value of 1 is equivalent to no skipping of files regardless of the file number.

  drift_e2ds_file_skip = 1

  Used in: cal_DRIFT-PEAK_E2DS_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DRIFT-PEAK_E2DS_spirou.main()
  Level: Public

- **Minimum box size for min max smoothing (drift_peak_minmax_boxsize)**

Defines the minimum size of the box used to get the minimum and maximum pixel values (specifically minimum pixel values). Each box (defined as the pixel position ±box size) is used to work out the background value for that pixel. Value must be an integer larger than zero and less than half the number of columns (x-dimension).

drift_peak_minmax_boxsize = 6

Used in:	cal_DRIFT-PEAK_E2DS_spirou
Defined in:	constants_SPIROU.txt
Called in:	cal_DRIFT-PEAK_E2DS_spirou.main()
Level:	Public

- **Image column (x-dim) border size (drift_peak_border_size)**

Defines the number of pixels on either side of an image that should not be used to find FP peaks. This size must be larger to or equal to {drift_peak_fpbox_size}, therefore the fit to an individual FP does not go off the edge of the image. Value must be an integer larger to or equal to {drift_peak_fpbox_size} and less than and less than half the number of columns (x-dimension).

drift_peak_border_size = 3

Used in:	cal_DRIFT-PEAK_E2DS_spirou
Defined in:	constants_SPIROU.txt
Called in:	SpirouDRS.spirouRV.spirouRV.create_drift_file()
Level:	Public

- **Box size for fitting individual FP peak. (drift_peak_fpbox_size)**

Defines the half-box size (i.e. central position ±box size) of the box used to fit an individual FP peak. This size must be large enough to fit a peak but not too large as to encompass multiple FP peaks. The value must be an integer larger than zero and smaller than or equal to {drift_peak_border_size} (to avoid fitting off the edges of the image).

drift_peak_fpbox_size = 3

Used in:	cal_DRIFT-PEAK_E2DS_spirou
Defined in:	constants_SPIROU.txt
Called in:	SpirouDRS.spirouRV.spirouRV.create_drift_file(), SpirouDRS.spirouRV.spirouRV.get_drift()
Level:	Public

- **Minimum sigma above median for valid peak (drift_peak_peak_sig_lim)**

  Defines the flux a valid peak must have in order to be recognized as a valid peak (before the peak fitting is done). If a peaks meaximum is below this threshold it will not be used as a valid peak in finding the drifts. Value is a dictionary containing keys equivalient to the lamp types (currently this is 'fp' and 'hc'. The values of each must be a float greater than 1 for above the median and, between zero and 1 for below the median).

  drift_peak_peak_sig_lim = `fp':1.0, 'hc':7.0'fp':1.0, 'hc':7.0`

  Used in: cal_DRIFT-PEAK_E2DS_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouRV.spirouRV.create_drift_file()
  Level: Public

- **Minimum spacing between valid peaks (drift_peak_inter_peak_spacing)**

  Defines the minimum spacing peaks must have (between neighbouring peaks) in order to recognized as valid peaks (before the peak fitting is done). If peak is closer than this sepration to a previous peak the peak will not be used as a valid peak in finding the drifts. Value must be an integer greater than zero.

  drift_peak_inter_peak_spacing = 5

  Used in: cal_DRIFT-PEAK_E2DS_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouRV.spirouRV.create_drift_file()
  Level: Public

- **Expected width of FP peaks (drift_peak_exp_width)**

  Defines the expected width of the FP peaks. Parameter is used to 'normalise' the peaks which are then subsequently removed if:

  $$\text{normalized FP FWHM} > \text{drift\_peak\_norm\_width\_cut} \tag{10.13}$$

  this is equivalent to:

  $$\text{FP FWHM} > (\text{drift\_peak\_exp\_width} + \text{drift\_peak\_norm\_width\_cut}) \tag{10.14}$$

  Value must be a float larger than zero and less than the number of columns (x-dimension).

  drift_peak_exp_width = 0.8

  Used in: cal_DRIFT-PEAK_E2DS_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouRV.spirouRV.remove_wide_peaks(), SpirouDRS.spirouRV.spirouRV.get_drift()
  Level: Public

- **Normalized FP width threshold (drift_peak_norm_width_cut)**

Defines the maximum 'normalized' width of FP peaks that is acceptable for a valid FP peak. i.e. widths above this threshold are rejected as valid FP peaks. This works as follows:

$$\text{normalized FP FWHM} > \text{drift\_peak\_norm\_width\_cut} \tag{10.15}$$

this is equivalent to:

$$\text{FP FWHM} > (\text{drift\_peak\_exp\_width} + \text{drift\_peak\_norm\_width\_cut}) \tag{10.16}$$

Value must be a float larger than zero and less than the number of columns (x-dimension) but if {drift_peak_exp_width} is defined sensibly then this number should be small.

drift_peak_norm_width_cut = 0.2

Used in:	cal_DRIFT-PEAK_E2DS_spirou
Defined in:	constants_SPIROU.txt
Called in:	SpirouDRS.spirouRV.spirouRV.remove_wide_peaks()
Level:	Public

- **Get drift via a Gaussian fitting process (drift_peak_getdrift_gaussfit)**

Defines whether the drift is calculated via a Gaussian fitting process (fitting the targeted order with a Gaussian) – $\sim \times 10$ slower, or adjusts a barycenter to get the drift. Value must be True or 1 to do the Gaussian fit, or False or 0 to use the barycenter adjustment.

drift_peak_getdrift_gaussfit = False

Used in:	cal_DRIFT-PEAK_E2DS_spirou
Defined in:	constants_SPIROU.txt
Called in:	cal_DRIFT-PEAK_E2DS_spirou.main()
Level:	Public

- **Pearson R coefficient (between reference and image) (drift_peak_pearsonr_cut)**

Defines the threshold below which a image is deemed to dissimilar from the reference image to be used. A Pearson R test is performed between the reference image (e2ds file) and the current iteration image (e2ds file), the minimum of all usable orders is then tested. If any order does not pass the criteria:

$$\text{coefficient}_{\text{order}} > \text{drift\_peak\_pearsonr\_cut} \tag{10.17}$$

then the whole image (e2ds file) is rejected. Value must be a float larger than zero and less than 1.0, values should be close to unity for a good fit i.e. 0.97.

drift_peak_pearsonr_cut = 0.9

Used in:	cal_DRIFT-PEAK_E2DS_spirou
Defined in:	constants_SPIROU.txt
Called in:	cal_DRIFT-PEAK_E2DS_spirou.main()
Level:	Public

Note: This value is currently set below a recommended level and should be set back to 0.97 as soon as possible, even coefficients at 0.95 are from very bad orders, and orders should be removed. A plot currently is made when a bad file is found (i.e. when the above cut is not met).

- **Sigma clip for found FP peaks (drift_peak_sigmaclip)**

  Defines the number of sigmas above the median that is used to remove bad FP peaks from the drift calculation process. Value must be a float larger than zero.

  drift_peak_sigmaclip = 1.0

  Used in: cal_DRIFT-PEAK_E2DS_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DRIFT-PEAK_E2DS_spirou.main()
  Level: Public

- **Plot linelist vs log Amplitude (drift_peak_plot_line_log_amp)**

  Defines whether we plot the line list against log amplitude. Value must be 1 or True to plot, or 0 or False to not plot

  drift_peak_plot_line_log_amp = False

  Used in: cal_DRIFT-PEAK_E2DS_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DRIFT-PEAK_E2DS_spirou.main()
  Level: Public

10.13 Bad pixel calibration variables

- **Bad pixel median image box width (badpix_flat_med_wid)**

  A similar flat is produced by taking the running median of the flat in the column direction (x-dimension) over a boxcar width of {badpix_flat_med_wid}. This assumes that the flux level varies only by a small amount over {badpix_flat_med_wid} pixels and that the bad pixels are isolated enough that the median along that box will be representative of the flux they should have if they were not bad. Value should be an integer larger than zero and less than the number of columns (x-axis dimension).

  badpix_flat_med_wid = 7

  Used in: cal_BADPIX_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouImage.spirouImage.normalise_median_flat(), SpirouDRS.spirouImage.spirouImage.locate_bad_pixels()
  Level: Public

  Note: Formally this was called wmed in cal_BADPIX_spirou

- **Bad pixel illumination cut parameter (badpix_illum_cut)**

  Threshold below which a pixel is considered unilluminated. As we cut the pixels that fractionally deviate by more than a certain amount ({badpix_flat_cut_ratio}) this would lead to lots of bad pixels in unilluminated regions of the array. This parameter stops this, as the pixels are normalised this value must be a float greater than zero and less than 1.

  badpix_illum_cut = 0.05

  Used in: cal_BADPIX_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouImage.spirouImage.locate_bad_pixels()
  Level: Public

  Note: Formally this was called illum_cut in cal_BADPIX_spirou

- **Bad pixel maximum differential pixel cut ratio (badpix_flat_cut_ratio)**

  This sets the maximum differential pixel response relative to the expected value. Value must be a float larger than zero.

  badpix_flat_cut_ratio = 0.5

  Used in: cal_BADPIX_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouImage.spirouImage.locate_bad_pixels()
  Level: Public

> **Note:** Formally this was called cut_ratio in cal_BADPIX_spirou

- **Bad pixel maximum flux to considered too hot (badpix_max_hotpix)**

  Defines the maximum flux value to be considered too hot to user.

  badpix_max_hotpix = 100.0

  Used in: cal_BADPIX_spirou
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouImage.spirouImage.locate_bad_pixels()
  Level: Public

> **Note:** Formally this was called max_hotpix in cal_BADPIX_spirou

10.14 Quality control variables

- **Maximum dark median level (qc_max_darklevel)**

  Defines the maximum dark median level in ADU/s. If this is greater than median flux it does not pass the quality control criteria:

  $$\text{Median Flux} < \text{qc\_max\_darklevel} \tag{10.18}$$

  Value must be a float equal to or larger than zero.

  qc_max_darklevel = 0.5

  Used in: cal_DARK_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DARK_spirou.main()
  Level: Public

- **Maximum percentage of dead pixels (qc_max_dead)**

  Defines the maximum allowed percentage of dead pixels in a dark image. If the number of dead pixels is greater than this it does not pass the quality control criteria:

  $$\text{dead pixels} = (\text{pixel value} > \text{dark\_cutlimit}) \text{ and } (\text{pixel value} \neq \text{NaN}) \tag{10.19}$$

  $$\text{Percentage of dead pixels} < \text{qc\_max\_dead} \tag{10.20}$$

  qc_max_dead = 20.0

  Used in: cal_DARK_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DARK_spirou.main()
  Level: Public

- **Maximum percentage of bad dark pixels (qc_max_dark)**

  Defines the maximum allowed percentage of bad dark pixels in a dark image. If the number of dead pixels is greater than this it does not pass the quality control criteria:

  $$\text{bad dark pixels} = \text{pixel value} > \text{dark\_cutlimit} \tag{10.21}$$

  $$\text{Percentage of bad dark pixels} < \text{qc\_max\_dead} \tag{10.22}$$

  qc_max_dark = 6.0

  Used in: cal_DARK_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DARK_spirou.main()
  Level: Public

- **Minimum dark exposure time (qc_dark_time)**

  Defines the minimum dark exposure time. If exposure time (from FITS rec HEADER) is below this the code will exit with 'Dark exposure time too short' message. Value must be a float greater than zero.

  qc_dark_time = 599.0

  Used in: cal_DARK_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_DARK_spirou.main()
  Level: Public

- **Maximum points removed in localization position fit (qc_loc_maxlocfit_removed_ctr)**

  Defines the maximum allowed number of points removed in the position fitting process (during localization). If number is more than this it does not pass the quality control criteria:

  $$\text{Number of rejected orders in center fit} > \text{qc\_loc\_maxlocfit\_removed\_ctr} \quad (10.23)$$

  Value must be a integer greater than zero.

  qc_loc_maxlocfit_removed_ctr = 1500

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.main()
  Level: Public

- **Maximum points removed in localization width fit (qc_loc_maxlocfit_removed_wid)**

  Defines the maximum allowed number of points removed in the width fitting process (during localization). If number is more than this it does not pass the quality control criteria:

  $$\text{Number of rejected orders in width fit} > \text{qc\_loc\_maxlocfit\_removed\_width} \quad (10.24)$$

  Value must be a integer greater than zero.

  qc_loc_maxlocfit_removed_wid = 105

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.main()
  Level: Public

- **Maximum allowed RMS in fitting in localization position fit (qc_loc_rmsmax_center)**

  Defines the maximum RMS allowed in the position fitting process (during localization). If the RMs is higher than this value it does not pass the quality control criteria:

  $$\text{Mean rms center fit} > \text{qc\_loc\_rmsmax\_center} \tag{10.25}$$

  Value must be a float greater than zero.

  qc_loc_rmsmax_center = 100

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.main()
  Level: Public

- **Maximum allowed RMS in fitting in localization width fit (qc_loc_rmsmax_fwhm)**

  Defines the maximum RMS allowed in the width fitting process (during localization). If the RMs is higher than this value it does not pass the quality control criteria:

  $$\text{Mean rms width fit} > \text{qc\_loc\_rmsmax\_fwhm} \tag{10.26}$$

  Value must be a float greater than zero.

  qc_loc_rmsmax_fwhm = 500

  Used in: cal_loc_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.main()
  Level: Public

- **Saturation point (qc_max_signal)**

  Defines the maximum signal allowed (when defining saturation limit). Currently this does not contribute to failing the quality test. Value must be a float greater than zero.

  qc_max_signal = 65500

  Used in: cal_FF_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_FF_RAW_spirou.main()
  Level: Public

Note: Currently this does not stop the file from passing the quality control criteria, it either should fail or should be removed.

- **Saturation level reached warning (qc_loc_flumax)**

  Defines the level above which a warning is generated in the form 'SATURATION LEVEL REACHED on Fiber {fiber}'. Value must be a float greater than zero.

  qc_loc_flumax = 64500

  Used in: cal_FF_RAW_spirou
  Defined in: constants_SPIROU.txt
  Called in: cal_FF_RAW_spirou.main()
  Level: Public

10.15 Calibration database variables

- **The calibration database master filename (ic_calibDB_filename)**

  Defines the name of the master calibration database text file for use in all calibration database operation.

  ic_calibDB_filename = master_calib_SPIROU.txt

  Used in: All Recipes
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouConfig.spirouConst.CALIBDB_MASTERFILE()
  Level: Public

  Note: This should probably be private, unless we want the user to be able to change calibDB files.

- **Maximum wait time for locked calibration database (calib_max_wait)**

  Defines the maximum time the code waits for the calibration database when it is locked. A locked file is created every time the calibration database is open (and subsequently closed when reading of the database was successful). If a lock file is present the code will wait a maximum of this many seconds and keep checking whether the lock file has been removed. After which time the code will exit with an error. Value must be a positive float greater than zero. Measured in seconds.

  calib_max_wait = 3600

  Used in: All Recipes
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouCDB.spirouCDB.get_check_lock_file()
  Level: Public

10.16 Startup variables

- **Configuration Folder Path (config_folder)**

  Defines the location of the configuration directory relative to the module directory (defined in variable = 'package'). Value must be a string containing a valid directory location.

  config_folder = ../config

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.CONFIGFOLDER()
  Called in: All Recipes
  Level: Private

- **Configuration file name (config_file)**

  Defines the main configuration (containing the data directories etc). Value must be a string containing a valid file name i.e. the main configuration file should be at {INSTALL_DIR}/{config_folder}/{config_file}.

  config_file = ../config/config.txt

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.CONFIGFILE()
  Called in: All Recipes
  Level: Private

- **Filenames from run time arguments (arg_file_names)**

  Gets the filenames from run time arguments.

  arg_file_names

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.ARG_FILE_NAMES()
  Called in: SpirouDRS.spirouStartup.spirouStartup.run_time_args()

- **Night name from run time arguments (arg_night_name)**

  Gets the night name from run time arguments.

  arg_night_name

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.ARG_NIGHT_NAME()
  Called in: SpirouDRS.spirouStartup.spirouStartup.run_time_args()

- **Calibration database file path (masterfilepath)**

  Gets the full calibration database file path

  masterfilepath

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.CALIBDB_MASTERFILE()
  Called in: SpirouDRS.spirouCDB.spirouCDB.write_files_to_master(), SpirouDRS.spirouCDB.spirouCDB.read_master_file(), SpirouDRS.spirouImage.spirouImage.correct_for_dark()

- **Calibration database lock file path (lockfilepath)**

  Gets the full calibration database lock file path

  lockfilepath

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.CALIBDB_LOCKFILE()
  Called in: SpirouDRS.spirouCDB.spirouCDB.get_check_lock_file()

- **Fits file name (fitsfilename)**

  Gets the full file path of the first file in 'arg_file_names'

  fitsfilename

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.FITSFILENAME()
  Called in: SpirouDRS.spirouStartup.spirouStartup.run_time_args()

- **Log program name (log_opt)**

  Chooses the display format for the program in the logging system.

  log_opt

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.LOG_OPT()
  Called in: SpirouDRS.spirouStartup.spirouStartup.run_time_args()

- **Documentation info manual file path (manual_file)**

  Gets the full documentation info manual file path

  manual_file

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.MANUAL_FILE()
  Called in: SpirouDRS.spirouStartup.spirouStartup.display_help_file()

- **Number of frames (nbframes)**

  Gets the number of frames from the list of files ('arg_file_names').

  nbframes

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.NBFRAMES
  Called in: SpirouDRS.spirouStartup.spirouStartup.run_time_args()

- **Program name from run time (program)**

  Gets the run program name from run time.

  program

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.PROGRAM()
  Called in: SpirouDRS.spirouStartup.spirouStartup.run_time_args()

- **Full path of raw data directory (raw_dir)**

  Gets the full path of the raw data directory.

  raw_dir

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.RAW_DIR()
  Called in: SpirouDRS.spirouImage.spirouFITS.math_controller(),
  SpirouDRS.spirouImage.spirouImage.get_all_similar_files(),
  SpirouDRS.spirouStartup.spirouStartup.display_run_files()

- **Full path of reduced data directory (reduced_dir)**

  Gets the full path of the reduced data directory.

  reduced_dir

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.REDUCED_DIR()
  Called in: SpirouDRS.spirouStartup.spirouStartup.run_time_args()

10.17 Formatting variables

- **Header date format (date_fmt_header)**

  Defines the format of the date in the FITS rec header files

  date_fmt_header = %Y-%m-%d-%H:%M:%S.%f

  Used in: SpirouDRS.spirouCDB.spirouCDB
  Defined in: SpirouDRS.spirouConfig.spirouConst.DATE_FMT_HEADER()
  Called in: SpirouDRS.spirouCDB.spirouCDB.update_database(), SpirouDRS.spirouCDB.spirouCDB.get_database()
  Level: Private

- **Calibration database date format (date_fmt_calibdb)**

  Defines the format of the date in the calibration database file

  date_fmt_calibdb = %Y-%m-%d-%H:%M:%S.%f

  Used in: SpirouDRS.spirouCDB.spirouCDB
  Defined in: SpirouDRS.spirouConfig.spirouConst.DATE_FMT_CALIBDB()
  Called in: SpirouDRS.spirouCDB.spirouCDB.update_database(), SpirouDRS.spirouCDB.spirouCDB.get_database()
  Level: Private

10.18 FITS rec variables

- **Forbidden copy keys**

  Lists the keys that should not be copied when call to copy all FITS rec keys is made. Should be a list of python strings.

```
forbidden_keys = ['SIMPLE', 'BITPIX', 'NAXIS', 'NAXIS1', 'NAXIS2', 'EXTEND',
                  'COMMENT', 'CRVAL1', 'CRPIX1', 'CDELT1', 'CRVAL2', 'CRPIX2',
                  'CDELT2', 'BSCALE', 'BZERO', 'PHOT_IM', 'FRAC_OBJ', 'FRAC_SKY',
                  'FRAC_BB']
```

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.FORBIDDEN_COPY_KEYS()
  Called in: SpirouDRS.spirouImage.spirouFITS
  Level: Private

10.19 Logging and printing variables

- **Print message level (PRINT_LEVEL)**

  The level of messages to print, values can be as follows:

  - "all" – prints all events
  - "info" – prints info, warning and error events
  - "warning" – prints warning and error events
  - "error" – print only error events

  Value must be a valid string. See section 7.2.2 for more details.

  PRINT_LEVEL = all

  Used in: All Recipes
  Defined in: ../config/config.txt
  Called in: SpirouDRS.spirouConfig.spirouConfig.check_params()
  Level: Public

- **Log message level (LOG_LEVEL)**

  The level of messages to print, values can be as follows:

  - "all" – prints all events
  - "info" – prints info, warning and error events
  - "warning" – prints warning and error events
  - "error" – print only error events

  Value must be a valid string. See section 7.2.2 for more details.

  LOG_LEVEL = all

  Used in: All Recipes
  Defined in: ../config/config.txt
  Called in: SpirouDRS.spirouConfig.spirouConfig.writelog()
  Level: Public

- **Logging keys (trig_key)**

Defines the logging keys to use for each logging levels. Value should be a dictionary of key value pairs (where all keys and values are strings). When using the {SpirouDRS.spirouCore.spirouLog.logger()} (aliases to {WLOG} or {wlog} in recipes) the first argument must be one of these keys and the returned string is the corresponding value. The keys of {write_level} and {trig_key} must be identical. See section 7.2.2 for more details.

trig_key = `dict(all=' ', error='!', warning='@', info='*', graph='~')`

Used in: All Recipes
Defined in: SpirouDRS.spirouConfig.spirouConst.LOG_TRIG_KEYS()
Called in: All Recipes
Level: Private

i.e.:

Python/Ipython

```
trig_key = dict(error='!')
WLOG('error', 'program', 'Message')
```

returns

Command line output

```
HH:MM:SS.s - ! |program|Message
```

- **Write level (write_level)**

Defines the write levels to use for each write level. A write level is defined by a number. The higher the number to more exclusive the level i.e. if A and B are write levels and A > B and write level is set to A, any log or print messages at level B will not be logged/printed. Printing is controlled by variable {PRINT_LEVEL} and logging by variable {LOG_LEVEL}. The keys of {write_level} and {trig_key} must be identical. See section 7.2.2 for more details.

write_level = `dict(error=3, warning=2, info=1, graph=0, all=0)`

Used in: All Recipes
Defined in: SpirouDRS.spirouConfig.spirouConst.LOG_TRIG_KEYS()
Called in: All Recipes
Level: Private

i.e.:

Python/Ipython

```
write_level = dict(error=3, warning=2, info=1)
trig_key = dict(all=' ', error='!', warning='@', info='*', graph='~')
PRINT_LEVEL = 'warning'

WLOG('error', 'program', 'Error message')
WLOG('warning', 'program', 'Warning message')
WLOG('info', 'program', 'Info message')
```

returns

Command line output

```
HH:MM:SS.s - ! |program|Error message
HH:MM:SS.s - @ |program|Warning message
```

Note: Note the info message was not shown as info=1 and {PRINT_LEVEL} is set to warning=2.

- **Logger exit type (log_exit_type)**

What to do when a logging 'error' is raise. Options are: 'None', 'os' or 'sys'. If 'None' the code continues on an 'error', if 'os' then python executes a 'os._exit' command (a hard exit), if 'sys' then python executes a 'sys.exit' command (a soft exit).

log_exit_type = sys

Used in: All Recipes
Defined in: SpirouDRS.spirouConfig.spirouConst.LOG_EXIT_TYPE()
Called in: SpirouDRS.spirouConfig.spirouConst.EXIT() which is called in SpirouDRS.spirouCore.spirouLog
Level: Private

- **Exit controller (exit)**

  Controls the exit type from 'log_exit_type' and {SpirouDRS.spirouConfig.spirouConst.LOG_EXIT_TYPE()}.

  exit

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.EXIT()
  Called in: SpirouDRS.spirouCore.spirouLog

- **Exit levels (exit_levels)**

  Controls which levels (defined in {write_level} and {trig_key}) will lead to the exit statement (given in {exit} and {log_exit_type}). Values must be a list of strings where each entry must be in {write_level} and {trig_key}.

  exit_levels

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.EXIT_LEVELS()
  Called in: SpirouDRS.spirouCore.spirouLog

- **Log caught warnings (log_caught_warnings)**

  If True or 1, then if warnings are passed to {SpirouDRS.spirouCore.spirouLog.warninglogger()} and there are warnings present, will attempt to log these warnings using the {SpirouDRS.spirouCore.spirouLog.logger} function. i.e. will print the warning to screen/log file depending on logging settings.

  log_caught_warnings = True

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.LOG_CAUGHT_WARNINGS()
  Called in: SpirouDRS.spirouCore.spirouLog
  Level: Private

- **Configuration key error message (cerrmsg)**

  Defines the message that is used when a configuration key is missing

  cerrmsg

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouConst.CONFIG_KEY_ERROR
  Called in: SpirouDRS.spirouCore.spirouLog.get_logfilepath()

Chapter 11

Output header keywords

Keywords are defined as a list of three strings, the first key is the HEADER key, the second is the HEADER value and the last is the HEADER comment i.e.

Python/Ipython

```
kw_KEYWORD = [key, value, comment]
```

and in a FITS rec would product the following:

Command line output

```
key     = value              / comment
```

To better understand the keywords in the DRS we have laid out each keyword in the following way:

- **Keyword title (kw_variable)**

  Description of the keyword

  kw_variable = ["HEADER key", "Default HEADER value", "HEADER comment"]

  HEADER file entry:

  HEADER key = Default HEADER value \ HEADER comment

  Used in: The recipe the keyword is used in
  Defined in: The place where the keyword is defined
  Called in: The code where the keyword is used.

Note: All keywords are (and should be) loaded into the main parameter dictionary 'p' in all recipes and thus are accessed via:

Python/Ipython

```
variable = p["kw_variable"]
```

11.1 Global keywords

- **DRS Version (kw_version)**

  The current name and version of the DRS

  kw_version = ["VERSION", "{DRS_NAME}_{DRS_VERSION}", "DRS version"]

  HEADER file entry:

  VERSION = {DRS_NAME}_{DRS_VERSION} \ DRS version

  Used in: All Recipes
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: SpirouDRS.spirouConfig.spirouKeywords

- **Root for localization keywords (kw_root_drs_loc)**

  The root (prefix) for localization keywords

  kw_root_drs_loc = LO

  Used in: cal_extract_RAW_spirou, SpirouDRS.spirouConfig.spirouKeywords
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_extract_RAW_spirou.main(), SpirouDRS.spirouConfig.spirouKeywords
  Level: Private

- **Root for flat field keywords (kw_root_drs_flat)**

  The root (prefix) for flat field keywords

  kw_root_drs_flat = FF

  Used in: SpirouDRS.spirouConfig.spirouKeywords
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: SpirouDRS.spirouConfig.spirouKeywords
  Level: Private

- **Root for HC keywords (kw_root_drs_hc)**

  The root (prefix) for the HC keywords

  kw_root_drs_hc = LMP

  Used in: SpirouDRS.spirouConfig.spirouKeywords
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: SpirouDRS.spirouConfig.spirouKeywords
  Level: Private

> **Note:** Not currently used

11.2 Dark calibration keywords

- **Fraction of dead pixels (kw_DARK_DEAD)**

  Percentage of dead pixels on image

  kw_DARK_DEAD = ["DADEAD", "0", "Fraction dead pixels [%]"]

  HEADER file entry:

  ```
  DADEAD  =  0  \  Fraction dead pixels [%]
  ```

  Used in: cal_DARK_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_DARK_spirou.main()

- **Median dark level (kw_DARK_MED)**

  Median dark level of the image in ADU/s

  kw_DARK_MED = ["DAMED", "0", "median dark level [ADU/s]"]

  HEADER file entry:

  ```
  DAMED  =  0  \  median dark level [ADU/s]
  ```

  Used in: cal_DARK_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_DARK_spirou.main()

- **Fraction of dead pixels (blue part) (kw_DARK_B_DEAD)**

  Percentage of dead pixels on image on the blue part of the image

  kw_DARK_B_DEAD = ["DABDEAD", "0", "Fraction dead pixels blue part [%]"]

  HEADER file entry:

  ```
  DABDEAD  =  0  \  Fraction dead pixels blue part [%]
  ```

  Used in: cal_DARK_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_DARK_spirou.main()

- **Median dark level (blue part) (kw_DARK_B_MED)**

Median dark level of the image in ADU/s on the blue part of the image

kw_DARK_B_MED = ["DABMED", "0", "median dark level blue part [ADU/s]"]

HEADER file entry:

```
DABMED  =  0  \  median dark level blue part [ADU/s]
```

Used in: cal_DARK_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_DARK_spirou.main()

- **Fraction of dead pixels (red part) (kw_DARK_R_DEAD)**

Percentage of dead pixels on image on the red part of the image

kw_DARK_R_DEAD = ["DARDEAD", "0", "Fraction dead pixels red part [%]"]

HEADER file entry:

```
DARDEAD  =  0  \  Fraction dead pixels red part [%]
```

Used in: cal_DARK_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_DARK_spirou.main()

- **Median dark level (red part) (kw_DARK_R_MED)**

Median dark level of the image in ADU/s on the red part of the image

kw_DARK_R_MED = ["DARMED", "0", "median dark level red part [ADU/s]"]

HEADER file entry:

```
DARMED  =  0  \  median dark level red part [ADU/s]
```

Used in: cal_DARK_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_DARK_spirou.main()

- **Dark level threshold (kw_DARK_CUT)**

The dark level threshold in ADU/s

kw_DARK_CUT = ["DACUT", "{dark_cutlimit}", "Threshold of dark level retain [ADU/s]"]

HEADER file entry:

```
DACUT    =    {dark_cutlimit}    \    Threshold of dark level retain [ADU/s]
```

Used in: cal_DARK_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_DARK_spirou.main()

11.3 Localization calibration keywords

- **Mean packground (kw_LOCO_BCKGRD)**

  The mean background of an image (as a percentage).

  kw_LOCO_BCKGRD = ["{kw_root_drs_loc}BCKGRD", "0", "mean background [%]"]

  HEADER file entry:

  ```
  {kw_root_drs_loc}BCKGRD  =  0  \  mean background [%]
  ```

  Used in: cal_loc_RAW_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_loc_RAW_spirou.main()

- **Image conversion factor (kw_CCD_CONAD)**

  Image conversion factor [e-/ADU]

  kw_CCD_CONAD = ["CONAD", "0", "CCD conv factor [e-/ADU]"]

  HEADER file entry:

  ```
  CONAD  =  0  \  CCD conv factor [e-/ADU]
  ```

  Used in: cal_loc_RAW_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_loc_RAW_spirou.main()

  Note: Currently not set

- **CCD Readout Noise (kw_CCD_SIGDET)**

  The image readout noise in e-

  kw_CCD_SIGDET = ["SIGDET", "0", "CCD Readout Noise [e-]"]

  HEADER file entry:

  ```
  SIGDET  =  0  \  CCD Readout Noise [e-]
  ```

  Used in: cal_loc_RAW_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_loc_RAW_spirou.main()

- **Coeffecients position fits for orders (kw_LOCO_CTR_COEFF)**

The coefficients of the position fits

kw_LOCO_CTR_COEFF = ["{kw_root_drs_loc}CTR", "0", "'Coeff center'"]

HEADER file entry:

```
{kw_root_drs_loc}CTR  =  0  \  'Coeff center'
```

Used in: cal_loc_RAW_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_loc_RAW_spirou.main()

- **Degree of the fitting polynomial for localization position (kw_LOCO_DEG_C)**

The fit degree used in the position fit during localization

kw_LOCO_DEG_C = ["{kw_root_drs_loc}DEGCTR", "{ic_locdfitc}", "degree fit ctr ord"]

HEADER file entry:

```
{kw_root_drs_loc}DEGCTR  =  {ic_locdfitc}  \  degree fit ctr ord
```

Used in: cal_loc_RAW_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_loc_RAW_spirou.main()

- **Degree of the fitting polynomial for localization width (kw_LOCO_DEG_W)**

The fit degree used in the width fit during localization

kw_LOCO_DEG_W = ["{kw_root_drs_loc}DEGFWH", "0", "degree fit width ord"]

HEADER file entry:

```
{kw_root_drs_loc}DEGFWH  =  0  \  degree fit width ord
```

Used in: cal_loc_RAW_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_loc_RAW_spirou.main()

- **Degree of the fitting polynomial for localization position error (kw_LOCO_DEG_E)**

The fit degree used in the position error fit during localization

kw_LOCO_DEG_E = ["{kw_root_drs_loc}DEGERR", "0", "degree fit profile error"]

HEADER file entry:

{kw_root_drs_loc}DEGERR = 0 \ degree fit profile error

Used in: cal_loc_RAW_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_loc_RAW_spirou.main()

Note: Currently not set

- **Delta width 3 convolve shape model (kw_LOCO_DELTA)**

The delta width used in pixels for the 3 convolve shape model

kw_LOCO_DELTA = ["{kw_root_drs_loc}PRODEL", "IC_LOC_DELTA_WIDTH", "param model 3gau"]

HEADER file entry:

{kw_root_drs_loc}PRODEL = IC_LOC_DELTA_WIDTH \ param model 3gau

Used in: cal_loc_RAW_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_loc_RAW_spirou.main()

- **Coefficients width fits for orders (kw_LOCO_FWHM_COEFF)**

The coefficients of the width fits

kw_LOCO_FWHM_COEFF = ["{kw_root_drs_loc}FW", "0", "'Coeff fwhm'"]

HEADER file entry:

{kw_root_drs_loc}FW = 0 \ 'Coeff fwhm'

Used in: cal_loc_RAW_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_loc_RAW_spirou.main()

- **Number of orders localized (kw_LOCO_NBO)**

The number of orders obtained during localization

kw_LOCO_NBO = ["{kw_root_drs_loc}NBO", "0", "nb orders localized"]

HEADER file entry:

```
{kw_root_drs_loc}NBO   =   0   \   nb orders localized
```

Used in: cal_loc_RAW_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_loc_RAW_spirou.main()

- **Max image flux (kw_LOC_MAXFLX)**

The maximum flux in the image in ADU

kw_LOC_MAXFLX = ["{kw_root_drs_loc}FLXMAX", "0", "max flux in order [ADU]"]

HEADER file entry:

```
{kw_root_drs_loc}FLXMAX   =   0   \   max flux in order [ADU]
```

Used in: cal_loc_RAW_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_loc_RAW_spirou.main()

- **Max removed points - position fit (kw_LOC_SMAXPTS_CTR)**

Maximum number of removed points allowed for location fit

kw_LOC_SMAXPTS_CTR = ["{kw_root_drs_loc}CTRMAX", "0", "max rm pts ctr"]

HEADER file entry:

```
{kw_root_drs_loc}CTRMAX   =   0   \   max rm pts ctr
```

Used in: cal_loc_RAW_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_loc_RAW_spirou.main()

- **Max removed points - width fit (kw_LOC_SMAXPTS_WID)**

  Maximum number of removed points allowed for width fit

  kw_LOC_SMAXPTS_WID = ["{kw_root_drs_loc}WIDMAX", "0", "max rm pts width"]

  HEADER file entry:

  {kw_root_drs_loc}WIDMAX = 0 \ max rm pts width

  Used in: cal_loc_RAW_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_loc_RAW_spirou.main()

  Note: Formally this was called 'kw_LOC_Smaxpts_width'

- **Maximum RMS position fit (kw_LOC_RMS_CTR)**

  Maximum rms allowed for location fit

  kw_LOC_RMS_CTR = ["{kw_root_drs_loc}RMSCTR", "0", "max rms ctr"]

  HEADER file entry:

  {kw_root_drs_loc}RMSCTR = 0 \ max rms ctr

  Used in: cal_loc_RAW_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_loc_RAW_spirou.main()

- **Maximum RMS width fit (kw_LOC_RMS_WID)**

  Maximum rms allowed for width fit

  kw_LOC_RMS_WID = ["{kw_root_drs_loc}RMSWID", "0", "max rms width"]

  HEADER file entry:

  {kw_root_drs_loc}RMSWID = 0 \ max rms width

  Used in: cal_loc_RAW_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_loc_RAW_spirou.main()

  Note: Formally this was called 'kw_LOC_rms_fwhm'

11.4 Slit calibration keywords

- **Tilt order prefix (kw_TILT)**

  Tilt order keyword prefix

  kw_TILT = ["{kw_root_drs_loc}TILT", "0", "Tilt order"]

  HEADER file entry:

  ```
  {kw_root_drs_loc}TILT  =  0  \  Tilt order
  ```

  Used in: cal_SLIT_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_SLIT_spirou.main()

11.5 Flat fielding calibration keywords

- **SNR (kw_EXTRA_SN)**

  Signal to noise ratio for order center

  kw_EXTRA_SN = ["EXTSN", "0", "S_N order center"]

  HEADER file entry:

  ```
  EXTSN  =  0  \  S_N order center
  ```

  Used in: cal_FF_RAW_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_FF_RAW_spirou.main()

- **Flat field RMS (kw_FLAT_RMS)**

  Flat field RMS for order

  kw_FLAT_RMS = ["{kw_root_drs_loc}RMS", "0", "FF RMS order"]

  HEADER file entry:

  ```
  {kw_root_drs_loc}RMS  =  0  \  FF RMS order
  ```

  Used in: cal_FF_RAW_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_FF_RAW_spirou.main()

11.6 Extraction calibration keywords

- **Localization filename (kw_LOCO_FILE)**

  localization file used in extraction process

  kw_LOCO_FILE = ["{kw_root_drs_loc}FILE", "0", "Localization file used"]

  HEADER file entry:

  ```
  {kw_root_drs_loc}FILE  =  0  \  Localization file used
  ```

  Used in: cal_extract_RAW_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_extract_RAW_spirou.main()

11.7 Bad pixel calibration keywords

- **Fraction of hot pixels (kw_BHOT)**

  The Fraction of hot pixels on dark image (as a percentage)

  kw_BHOT = ["BHOT", "0", "Frac of hot px [%]"]

  HEADER file entry:

  ```
  BHOT  =  0  \  Frac of hot px [%]
  ```

  Used in: cal_BADPIX_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_BADPIX_spirou.main()

- **Fraction of bad pixels from flat (kw_BBFLAT)**

  The Fraction of bad pixels from flat image (as a percentage)

  kw_BBFLAT = ["BBFLAT", "0", "Frac of bad px from flat [%]"]

  HEADER file entry:

  ```
  BBFLAT  =  0  \  Frac of bad px from flat [%]
  ```

  Used in: cal_BADPIX_spirou
  Defined in: SpirouDRS.spirouConfig.spirouKeywords
  Called in: cal_BADPIX_spirou.main()

- **Fraction of non-finite pixels from dark (kw_BNDARK)**

The Fraction of non-finite pixels from dark image (as a percentage)

kw_BNDARK = ["BNDARK", "0", "Frac of non-finite px in dark [%]"]

HEADER file entry:

```
BNDARK  =  0  \  Frac of non-finite px in dark [%]
```

Used in: cal_BADPIX_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_BADPIX_spirou.main()

- **Fraction of non-finite pixels from flat (kw_BNFLAT)**

The Fraction of non-finite pixels from flat image (as a percentage)

kw_BNFLAT = ["BNFLAT", "0", "Frac of non-finite px in flat [%]"]

HEADER file entry:

```
BNFLAT  =  0  \  Frac of non-finite px in flat [%]
```

Used in: cal_BADPIX_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_BADPIX_spirou.main()

- **Fraction of bad pixels (kw_BBAD)**

The Fraction of bad pixels conforming to all criteria (as a percentage)

kw_BBAD = ["BBAD", "0", "Frac of bad px with all criteria [%]"]

HEADER file entry:

```
BBAD  =  0  \  Frac of bad px with all criteria [%]
```

Used in: cal_BADPIX_spirou
Defined in: SpirouDRS.spirouConfig.spirouKeywords
Called in: cal_BADPIX_spirou.main()

Chapter 12

The Recipes

12.1 The cal_DARK recipe

12.2 The cal_loc recipe

12.3 The cal_SLIT recipe

12.4 The cal_FF recipe

12.5 The cal_extract recipes

12.6 The cal_DRIFT recipes

12.7 The cal_BADPIX recipe

12.8 The cal_HC recipe

12.9 The cal_WAVE recipe

12.10 The cal_CCF recipe

12.11 The pol_spirou recipe

Chapter 13

The DRS Module

13.1 The spirouBACK module

13.2 The spirouCDB module

13.3 The spirouCore module

13.4 The spirouEXTOR module

13.5 The spirouFLAT module

13.6 The spirouImage module

13.7 The spirouLOCOR module

13.8 The spirouRV module

13.9 The spirouStartup module

13.10 The spirouUnitTests module