

SPIRou Data Reduction Software

User Guide

0.0.37

For DRS SPIRou 0.1.017 (alpha pre-release)

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

2018-02-13



Abstract

This is the guide to installing, running, and using the SPIRou DRS.

Contents

1	Introduction	1
2	Quick Install Guide	3
2.1	Linux	3
2.2	Mac	4
2.3	Windows	5
3	Installation	6
3.1	Introduction	6
3.2	Download	6
3.3	Prerequisites	7
3.3.1	Anaconda python distribution	7
3.3.2	Separate python installation	7
3.4	Installation Linux and macOS	8
3.4.1	Extraction	8
3.4.2	Modify environmental settings	8
3.4.3	Make recipes executable	8
3.5	Setting up the DRS on Linux and macOS	9
3.6	Validating Installation on Linux and macOS	10
3.7	Installation Windows	11
3.7.1	How to modify environmental settings in windows	11
3.8	Setting up the DRS	13
3.9	Validating Installation on Windows	14
4	Data Architecture	15
4.1	Installed file structure	15
4.2	The Installation root directory	16
4.2.1	The bin directory	16
4.3	The data root directory	17
4.3.1	The raw and reduced data directories	17
4.4	The calibration database directory	18
5	Using the DRS	19
5.1	Running the DRS recipes directly	19
5.2	Running the DRS recipes from a python script	19
5.3	Working example of the code for SPIrou	20
5.3.1	Overview	20
5.3.2	Run through from command line/python shell (Linux and macOS)	20
5.3.3	Run through python script	23
6	Required input header keywords	24
6.1	Required keywords	24
7	User modifiable variables	26
7.1	Variable file locations	26
7.2	Global variables	26
7.3	Directory variables	27
7.4	Image variables	29
7.5	Fiber variables	31
7.6	Dark calibration variables	34

7.7	Localization calibration variables	36
7.8	Slit calibration variables	42
7.9	Flat fielding calibration variables	43
7.10	Extraction calibration variables	45
7.11	Drift calibration variables	47
7.12	Drift-Peak calibration variables	51
7.13	Bad pixel calibration variables	55
7.14	Quality control variables	57
7.15	Calibration database variables	61
7.16	Logging and printing variables	62
8	The Recipes	63
8.1	The cal_DARK recipe	63
8.1.1	The inputs	63
8.1.2	The outputs	63
8.1.3	Summary of procedure	64
8.1.4	Quality Control	64
8.1.5	Example working run	65
8.1.6	Interactive mode	66
8.2	The cal_BADPIX recipe	67
8.2.1	The inputs	67
8.2.2	The outputs	67
8.2.3	Summary of procedure	67
8.2.4	Quality Control	68
8.2.5	Example working run	69
8.3	The cal_loc recipe	70
8.3.1	The inputs	70
8.3.2	The outputs	70
8.3.3	Summary of procedure	71
8.3.4	Quality Control	71
8.3.5	Example working run	72
8.4	The cal_SLIT recipe	75
8.5	The cal_FF recipe	76
8.6	The cal_extract recipes	77
8.7	The cal_DRIFT recipes	78
8.8	The cal_HC recipe	79
8.9	The cal_WAVE recipe	80
8.10	The cal_CCF recipe	81
8.11	The pol_spirou recipe	82
8.12	The validation recipes recipe	83

Chapter 1

Introduction

This documentation will cover the installation, data architecture, , using the DRS (with a working example), descriptions of the variables , and the recipes .

Variables are defined in detail in section 7 and will be defined throughout via the following syntax: **VARIABLE**. When referred to, one should take it as using the value set in section 7 by default or in the file described in the variables description 'Defined in' section. Clicking these variables will go to the appropriate variable description.

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.

```
Generic text file

# 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:

```
For example in ~/.bashrc

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

```
For example in ~/.tcshrc

#!/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

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

The following denotes a command line print out

```
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/Ipynb

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

Chapter 2

Quick Install Guide

2.1 Linux

This is a quick guide to installation, for a more full description please see Chapter 3 . This assumes you have the latest version of Anaconda for python 3 (or python 2) and are using BASH.

1. Get the latest version of the DRS (for SPIROU version 0.1.017 (alpha pre-release)). from here: https://github.com/njcuk9999/spirou_py3
2. Download the test data from here: http://genesis.astro.umontreal.ca/neil/spirou_test_data_alpha0.1.003.zip (if required).
3. Extract the DRS (make a note of the path, hereinafter `DRS_ROOT`)
4. Add the following paths to your PATH and PYTHON PATH environmental variables (in for example `.bashrc`)

e.g. in `~/.bashrc`

```
export PATH="DRS_ROOT/bin/:<$PATH>"
export PYTHONPATH="DRS_ROOT:DRS_ROOT/bin/:<$PYTHONPATH>"
```

5. make sure your paths are set

```
>> source ~/.bashrc
>> echo $PATH
```

6. Make recipes executable (found in the `DRS_ROOT/bin` folder) - to use from the command line.
7. Setup the DRS paths (edit the file: `'../config /config.txt'`):

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

8. validate the DRS installation:

```
>> cal_validate_spirou
```

or

```
>> python cal_validate_spirou
```

The DRS is now installed and setup. To run see section 5 .

2.2 Mac

This is a quick guide to installation, for a more full description please see Chapter 3 . This assumes you have the latest version of Anaconda for python 3 (or python 2) and are using BASH.

1. Get the latest version of the DRS (for SPIRou version 0.1.017 (alpha pre-release)). from here: https://github.com/njcuk9999/spirou_py3
2. Download the test data from here: http://genesis.astro.umontreal.ca/neil/spirou_test_data_alpha0.1.003.zip (if required).
3. Extract the DRS (make a note of the path, hereinafter `DRS_ROOT`)
4. Add the following paths to your PATH and PYTHON PATH environmental variables (in for example `.bashrc`)

e.g. in `~/ .bashrc`

```
export PATH="$DRS_ROOT/bin/:$PATH"
export PYTHONPATH="$DRS_ROOT:$DRS_ROOT/bin/:$PYTHONPATH"
```

5. make sure your paths are set

```
>> source ~/.bashrc
>> echo $PATH
```

6. Make recipes executable (found in the `DRS_ROOT/bin` folder) - to use from the command line.
7. Setup the DRS paths (edit the file: `'../config /config.txt'`):

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

8. validate the DRS installation:

```
>> cal_validate_spirou
```

or

```
>> python cal_validate_spirou
```

The DRS is now installed and setup. To run see section 5 .

2.3 Windows

This is a quick guide to installation, for a more full description please see Chapter 3 . This assumes you have the latest version of Anaconda for python 3 (or python 2)

1. Get the latest version of the DRS (for SPIROU version 0.1.017 (alpha pre-release)). from here: https://github.com/njcuk9999/spirou_py3
2. Download the test data from here: http://genesis.astro.umontreal.ca/neil/spirou_test_data_alpha0.1.003.zip (if required).
3. Extract the DRS (make a note of the path, hereinafter `DRS_ROOT`)
4. Add the following paths to your PATH environmental variable

In "Enviromental Variables"

```
DRS_ROOT\bin\;
```

5. Add the following paths to your PYTHONPATH environmental variable

In "Enviromental Variables"

```
%PYTHONPATH%;DRS_ROOT;DRS_ROOT\bin\;
```

6. Setup the DRS paths (edit the file: `../config /config.txt`):

<code>TDATA</code>	<code>= C:\\Users\\User\\Documents\\drs\\data</code>	/ Define the DATA directory
<code>DRS_ROOT</code>	<code>= C:\\Users\\User\\Documents\\drs\\INTROOT</code>	/ Define the installation directory
<code>DRS_DATA_RAW</code>	<code>= C:\\Users\\User\\Documents\\drs\\data\\raw</code>	/ Define the folder with the raw data files in
<code>DRS_DATA_REduc</code>	<code>= C:\\Users\\User\\Documents\\drs\\data\\reduced</code>	/ Define the directory that the reduced data should be saved to/read from
<code>DRS_CALIB_DB</code>	<code>= C:\\Users\\User\\Documents\\drs\\data\\calibDB</code>	/ Define the directory that the calibration files should be saved to/read from
<code>DRS_DATA_MSG</code>	<code>= C:\\Users\\User\\Documents\\drs\\data\\msg</code>	/ Define the directory that the log messages are stored in
<code>DRS_DATA_WORKING</code>	<code>= C:\\Users\\User\\Documents\\drs\\data\\tmp</code>	/ Define the working directory

Note: Note paths in windows must have a `'\\'` also the python files must be open with a valid editor such as sublime text, notepad++, spyder or pycharm for example

7. validate the DRS installation:

```
>> python cal_validate_spirou
```

The DRS is now installed and setup. To run see section 5 .

Chapter 3

Installation

3.1 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.

3.2 Download

Get the latest version of the DRS (for SPIROU version 0.1.017 (alpha pre-release)). Use any of the following ways:

- manually download from here: https://github.com/njcuk9999/spirou_py3\protect\kern+.1667em\relax

- use Git:

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

- use SVN:

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

- use ssh:

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

3.3 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)

3.3.1 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

3.3.2 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)
 - MATPLOTLIB (tested with version 2.0)
 - NUMPY (tested with version 1.12)
 - SCIPY (tested with version 0.19)
 - 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 1.2)
 - matplotlib (tested with version 2.1)
 - numpy (tested with version 1.13)
 - scipy (tested with version 1.0)
 - and the following built-in modules (comes with python): __FUTURE__, COLLECTIONS, DATE-TIME, FILECMP, GLOB, OS, PKG_RESOURCES, SHUTIL, SYS, TIME, WARNINGS

3.4 Installation Linux and macOS

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

1. Extraction (Section 3.4.1)
2. Modify environmental settings (Section 3.4.2)
3. Make recipes executable (Section 3.4.3)

3.4.1 Extraction

The first step is to extract the DRS into a folder (the `DRS_ROOT`).

Do this by using the following commands:

```
>> cd DRS_ROOT
>> unzip DRS.zip
```

3.4.2 Modify environmental settings

The next step is to modify your `PATH` and `PYTHONPATH` environmental variables (to include the `DRS_ROOT`). 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)

e.g. in `~/ .bashrc`

```
export PATH="DRS_ROOT/bin/:<$PATH>"
export PYTHONPATH="DRS_ROOT:DRS_ROOT/bin/:<$PYTHONPATH>"
```

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

e.g. in `~/ .tcshrc`

```
setenv PATH "DRS_ROOT/bin/":${PATH}
@setenv@ <PYTHONPATH> "DRS_ROOT:DRS_ROOT/bin/":${PYTHONPATH}
```

3.4.3 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:

```
>> chmod +x DRS_ROOT/bin/*.py
```

3.5 Setting up the DRS on Linux and macOS

Before running the DRS one must set the data paths.

The ‘../config /config.txt’ file is located in the `DRS_ROOT` in the config folder.
i.e. at `DRS_ROOT /config /config.txt`

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

<code>TDATA</code>	=	<code>/drs/data/</code>	/	Define the DATA directory
<code>DRS_ROOT</code>	=	<code>/drs/INTROOT/</code>	/	Define the installation directory
<code>DRS_DATA_RAW</code>	=	<code>/drs/data/raw</code>	/	Define the folder with the raw data files in
<code>DRS_DATA_REduc</code>	=	<code>/drs/data/reduced</code>	/	Define the directory that the reduced data should be saved to/read from
<code>DRS_CALIB_DB</code>	=	<code>/drs/data/calibDB</code>	/	Define the directory that the calibration files should be saved to/read from
<code>DRS_DATA_MSG</code>	=	<code>/drs/data/msg</code>	/	Define the directory that the log messages are stored in
<code>DRS_DATA_WORKING</code>	=	<code>/drs/data/tmp/</code>	/	Define the working directory

The directories here are for linux and macOS systems another example would be ‘/home/user/INTROOT’ for the `DRS_ROOT` directory.
On Windows machines this would be equivalent to ‘C:\\Users\\User\\Documents\\drs\\’.

Note: Note: On windows paths in windows must have a ‘\\’ also the python files must be open with a valid editor such as sublime text, notepad++, spyder or pycharm for example

The following keywords can be changed:

<code>DRS_PLOT</code>	=	1	/	Whether to show plots
<code>PRINT_LEVEL</code>	=	"all"	/	Level at which to print
<code>LOG_LEVEL</code>	=	"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

3.6 Validating Installation on Linux and macOS

Note: One must install the DRS (Section 3.4) AND set up the DRS (Section 3.8) 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:

```
>> cal_validate_spirou
```

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

```
>> 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:

```
HH:MM:SS.S - || *****
HH:MM:SS.S - || * SPIROU Q(#) Geneva Observatory (0.0.1)
HH:MM:SS.S - || *****
HH:MM:SS.S - ||(dir_data_raw)      DRS_DATA_RAW=/drs/data/raw
HH:MM:SS.S - ||(dir_data_reduc)    DRS_DATA_REDUCE=/drs/data/reduced
HH:MM:SS.S - ||(dir_calib_db)      DRS_CALIB_DB=/drs/data/calibDB
HH:MM:SS.S - ||(dir_data_msg)      DRS_DATA_MSG=/drs/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=/drs/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.
```

3.7 Installation Windows


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

1. Extract to `DRS_ROOT` with your favourite unzipping software.
2. Add `DRS_ROOT` to your PYTHONPATH (Section 3.7.1)

3.7.1 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'. Note in windows 10 you can also click the windows icon and type 'Advanced System Settings' then click 'Environment Variables'.
2. under system variable 'Path' click edit and add:



In "Enviromental Variables"

```
DRS_ROOT ;DRS_ROOT\bin;
```

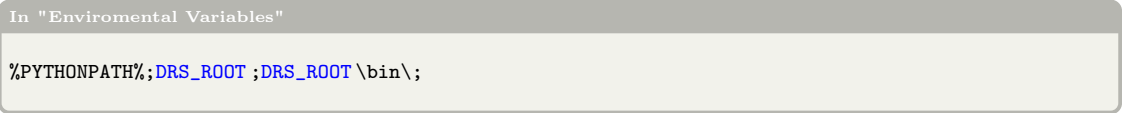
3. if under system variable 'PYTHONPATH' exists click edit and add '`DRS_ROOT`;' to the end.
i.e.



In "Enviromental Variables"

```
DRS_ROOT ;DRS_ROOT\bin;
```

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



In "Enviromental Variables"

```
%PYTHONPATH%;DRS_ROOT ;DRS_ROOT\bin\;
```

Figure 3.1 shows screengrabs of the various steps above to aid in updating PATH and PYTHONPATH. For problems/troubleshooting see here: <https://stackoverflow.com/questions/3701646>.

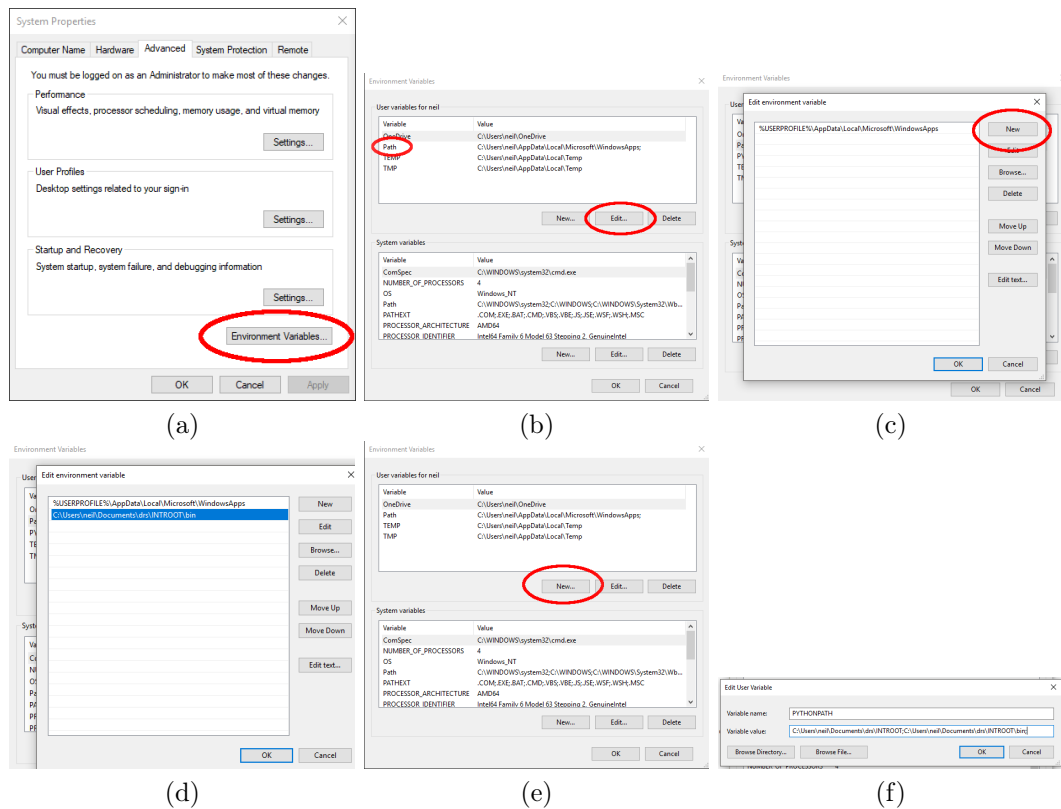


Figure 3.1: (a) Once in Advanced system properties click “Environment Variables” (b) Click ‘Path’ and click ‘Edit...’ to edit the ‘Path’ environmental variable (c) Once in the Path environmental variable click “New” to add a new path (d) Type in the new line to add variable and click “OK” (e) Once back in the Environmental variable page click “New” to add ‘PYTHONPATH’ (f) Set the variable name to “PYTHONPATH” and edit the variable value accordingly.

3.8 Setting up the DRS

Before running the DRS one must set the data paths.

The ‘../config /config.txt’ file is located in the `DRS_ROOT` in the config folder.
i.e. at `DRS_ROOT\\config\\config.txt`

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

<code>TDATA</code>	<code>= C:\\Users\\User\\ \\Documents\\drs\\ \\data</code>	/ Define the DATA directory
<code>DRS_ROOT</code>	<code>= C:\\Users\\User\\ \\Documents\\drs\\ \\INTROOT</code>	/ Define the installation directory
<code>DRS_DATA_RAW</code>	<code>= C:\\Users\\User\\ \\Documents\\drs\\ \\data\\raw</code>	/ Define the folder with the raw data files in
<code>DRS_DATA_REDUC</code>	<code>= C:\\Users\\User\\ \\Documents\\drs\\ \\data\\reduced</code>	/ Define the directory that the reduced data should be saved to/read from
<code>DRS_CALIB_DB</code>	<code>= C:\\Users\\User\\ \\Documents\\drs\\ \\data\\calibDB</code>	/ Define the directory that the calibration files should be saved to/read from
<code>DRS_DATA_MSG</code>	<code>= C:\\Users\\User\\ \\Documents\\drs\\ \\data\\msg</code>	/ Define the directory that the log messages are stored in
<code>DRS_DATA_WORKING</code>	<code>= C:\\Users\\User\\ \\Documents\\drs\\ \\data\\tmp</code>	/ Define the working directory

Note: Note: On windows paths in windows must have a ‘\\’ also the python files must be open with a valid editor such as sublime text, notepad++, spyder or pycharm for example

The following keywords can be changed:

<code>DRS_PLOT</code>	<code>= 1</code>	/ Whether to show plots
<code>PRINT_LEVEL</code>	<code>= "all"</code>	/ Level at which to print
<code>LOG_LEVEL</code>	<code>= "all"</code>	/ 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

3.9 Validating Installation on Windows

Note: One must install the DRS (Section 3.7) AND set up the DRS (Section 3.8) 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:

```
>> 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:

```
17:34:19.0 - || *****
17:34:19.0 - || * SPIROU @(#) Geneva Observatory (0.1.016)
17:34:19.0 - || *****
17:34:19.0 - ||(dir_data_raw)      DRS_DATA_RAW=C:\\Users\\User\\Documents\\drs\\data\\raw
17:34:19.0 - ||(dir_data_reduc)    DRS_DATA_REDUCE=C:\\Users\\User\\Documents\\drs\\data\\reduced
17:34:19.0 - ||(dir_calib_db)      DRS_CALIB_DB=C:\\Users\\User\\Documents\\drs\\data\\calibDB
17:34:19.0 - ||(dir_data_msg)      DRS_DATA_MSG=C:\\Users\\User\\Documents\\drs\\data\\msg
17:34:19.0 - ||(print_level)      PRINT_LEVEL=all          %(error/warning/info/all)
17:34:19.0 - ||(log_level)         LOG_LEVEL=all          %(error/warning/info/all)
17:34:19.0 - ||(plot_graph)       DRS_PLOT=1            %(def/undef/trigger)
17:34:19.0 - ||(used_date)        DRS_USED_DATE=undefined
17:34:19.0 - ||(working_dir)      DRS_DATA_WORKING=C:\\Users\\User\\Documents\\drs\\data\\tmp
17:34:19.0 - ||                  DRS_INTERACTIVE is not set, running on-line mode
17:34:19.0 - ||                  DRS_DEBUG is set, debug mode level:1
17:34:19.0 - ||
17:34:19.0 - ||Validation successful. DRS installed corrected.
```

Chapter 4

Data Architecture

Described below is the file structure, after correct installation (Chapter 3).

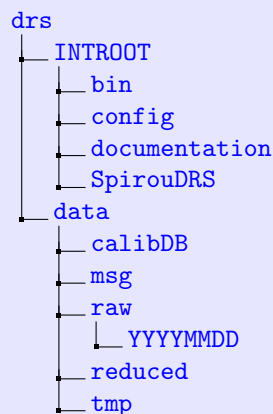
4.1 Installed file structure

The file structure should look as follows:



* This is the recommended file structure and raw, reduced, calibDB, msg and tmp can be changed using the `DRS_DATA_RAW`, `DRS_DATA_REduc`, `DRS_CALIB_DB`, `DRS_DATA_MSG`, and `DRS_DATA_WORKING` variables in Section 3.8.

i.e. for the paths given in Section 3.8 this would be:



4.2 The Installation root directory

The `DRS_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
```

4.2.1 The bin directory

The bin directory is located in the `DRS_ROOT` directory. This contains all the recipes that can be used. A detailed description of all recipes can be found in Chapter 8 but are listed here for completeness.

```
{dir}
├── {DRS_ROOT}
│   ├── bin .....Recipes
│   │   ├── cal_BADPIX_spirou .....See Section 8.2
│   │   ├── cal_CCF_E2DS_spirou .....See Section 8.10
│   │   ├── cal_DARK_spirou .....See Section 8.1
│   │   ├── cal_DRIFT_RAW_spirou .....See Section 8.7
│   │   ├── cal_DRIFT_E2DS_spirou .....See Section 8.7
│   │   ├── cal_DRIFT-PEAK_E2DS_spirou .....See Section 8.7
│   │   ├── cal_extract_RAW_spirou .....See Section 8.6
│   │   ├── cal_extract_RAW_spirouAB .....See Section 8.6
│   │   ├── cal_extract_RAW_spirouC .....See Section 8.6
│   │   ├── cal_FF_RAW_spirou .....See Section 8.5
│   │   ├── cal_HC_E2DS_spirou .....See Section 8.8
│   │   ├── cal_loc_RAW_spirou .....See Section 8.3
│   │   ├── cal_SLIT_spirou .....See Section 8.4
│   │   ├── cal_validate_spirou .....See Section 8.12
│   │   └── cal_WAVE_E2DS_spirou .....See Section 8.9
```

4.3 The data root directory

This is the directory where all the data should be stored. The default and recommended design is to have `DRS_DATA_RAW`, `DRS_DATA_REDUCE`, `DRS_CALIB_DB`, `DRS_DATA_MSG`, and `DRS_DATA_WORKING` as sub-directories of `DRS_ROOT`. However as in Section 3.8. these sub-directories can be defined elsewhere.

4.3.1 The raw and reduced data directories

The raw observed data is stored under the `DRS_DATA_RAW` path, the files are stored by night in the form `YYYYMMDD`.

The file structure can be seen below:

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

4.4 The calibration database directory

```
{TDATA}
├─ calibDB or {DRS_CALIB_DB}
│   └─ master_calib_SPIROU.txt
│       └─ .....The calibration fits files
```

The calibDB contains all the calibration files that pass the quality tests and a test file `ic_calibDB_filename`. It is located at `DRS_CALIB_DB` or if this is not defined is located by default at the `TDATA` directory. Each line in this file is a unique calibration file and lines are formatted in the following manner:

In calibration database file

```
{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 `DRS_DATA_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 `ic_calibDB_filename` is shown below (assuming the listed files are present in `DRS_CALIB_DB`)

In calibration database file

```
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 5

Using the DRS

There are two ways to run the DRS recipes. The first (described in Section 5.1) 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 5.2).

5.1 Running the DRS recipes directly

As in Chapter 3, 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 scripts. Below we use `cal_DARK_spirou` as an example:

- To run from command line type:

```
>> cal_DARK_spirou YYMMDD Filenames
```

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

```
>> python cal_DARK_spirou YYMMDD Filenames
>> ipython cal_DARK_spirou YYMMDD Filenames
```

- To run from ipython, open ipython and type:

Python/Ipypthon

```
run cal_DARK_spirou YYMMDD Filenames
```

5.2 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_spirou` as an example:

Python/Ipypthon

```
# 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)
```

5.3 Working example of the code for SPIRou

5.3.1 Overview

For this example all files are from:

```
>> 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 3.8 and shown explicitly in Section 4.1) all files should be places in the [DRS_DATA_RAW](#) ([/drs/data/raw](#) in our case).

and we will also need the current WAVE file from here:

```
>> 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 8.1)
2. [cal_loc_RAW_spirou](#) ($\times 2$) (See Section 8.3)
3. [cal_SLIT_spirou](#) (See Section 8.4)
4. [cal_FF_RAW_spirou](#) ($\times 2$) (See Section 8.5)
5. (add [spirou_wave_ini3.fits](#) to [calibDB](#))
6. [cal_extract_RAW_spirouAB](#) and [cal_extract_RAW_spirouC](#) (many times) (See Section 8.6)
7. [cal_DRIFT_RAW_spirou](#) (See Section 8.7)

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

As long as all codes are excutable (see Section 3.4.3) 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:

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

2. run the order localisation on the ‘dark_flat’ files:

```
>> 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:

```
>> 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.

```
>> 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 AND `calib_db_match="older"` you will get an error message when running the file (the tilt file is newer than input data) to solve this change `calib_db_match="closest"`.

```
>> 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 ones from here:

```
>> spirou@10.102.14.81:/data/reduced/DATA-CALIB/spirou_wave_ini3.fits
>> spirou@10.102.14.81:/data/.../AT4-04/2017-10-11_21-32-17 2017
    -10-11_21-32-17_hcone_hcone02c406_wave_AB.fits
>> spirou@10.102.14.81:/data/.../AT4-04/2017-10-11_21-32-17 2017
    -10-11_21-32-17_hcone_hcone02c406_wave_C.fits
```

then place it in the `DRS_CALIB_DB` folder. You will also need to edit the '`ic_calibDB_filename`' file located in `DRS_CALIB_DB`.

Add the following line to '`ic_calibDB_filename`'

In calibration database file

```
WAVE_AB AT4-04/2017-10-11_21-32-17 2017-10-11_21-32-17_hcone_hcone02c406_wave_AB.fits
    2014-08-11-03:42:12.000000 1407728532
WAVE_A 20170710 spirou_wave_ini3.fits 2000-01-01-00:00:00.00 946684800
WAVE_B 20170710 spirou_wave_ini3.fits 2000-01-01-00:00:00.00 946684800
WAVE_C AT4-04/2017-10-11_21-32-17 2017-10-11_21-32-17_hcone_hcone02c406_wave_C.fits
```

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

In calibration database file

```
WAVE_AB AT4-04/2017-10-11_21-32-17 2017-10-11_21-32-17_hcone_hcone02c406_wave_AB.fits
    2014-08-11-03:42:12.000000 1407728532
WAVE_A 20170710 spirou_wave_ini3.fits 2000-01-01-00:00:00.00 946684800
WAVE_B 20170710 spirou_wave_ini3.fits 2000-01-01-00:00:00.00 946684800
WAVE_C AT4-04/2017-10-11_21-32-17 2017-10-11_21-32-17_hcone_hcone02c406_wave_C.fits
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
```

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:

```
>> 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:

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

5.3.3 Run through python script

The process is in the same order as Section 5.3.2, including adding the ‘WAVE’ lines to the calibDB folder).

Python/Ipypthon

```
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 6

Required input header keywords

6.1 Required keywords

The following keywords are required by the current recipes to run.

- **Data fits file type (`kw_DPRTYPE`)**

The data fits file type (template Name)

```
kw_DPRTYPE = ["TPL_NAME", "DATA", "template Name"]
```

HEADER file entry:

```
TPL_NAME = "DATA" \ template Name
```

Used in: All Recipes

Defined in: SpirouDRS.spirouConfig.spirouKeywords

- **Acquisition time (human readable) (`kw_ACQTIME_KEY`)**

The acquisition time in format YYYY-mm-dd-HH-MM-SS.ss

```
kw_ACQTIME_KEY = ["ACQTIME1", "YYYY-mm-dd-HH-MM-SS.ss", "Date at start of observation"]
```

HEADER file entry:

```
ACQTIME1 = YYYY-mm-dd-HH-MM-SS.ss \ Date at start of observation
```

Used in: All Recipes

Defined in: SpirouDRS.spirouConfig.spirouKeywords

- **Acquisition time (unix time format) (`kw_ACQTIME_KEY_UNIX`)**

The acquisition time in in unix time format (time since 1970-01-01-00-00-00)

```
kw_ACQTIME_KEY_UNIX = ["ACQTIME", "000000000.00", "Date in unix time at start of observation"]
```

HEADER file entry:

```
ACQTIME = 000000000.00 \ Date in unix time at start of observation
```

Used in: All Recipes

Defined in: SpirouDRS.spirouConfig.spirouKeywords

- **Read noise (`kw_RDNOISE`)**

The read noise (used for sigdet) [e-]

```
kw_RDNOISE = ["RDNOISE", "0.0", "read noise (electrons)"]
```

HEADER file entry:

```
RDNOISE    = 0.0 \ read noise (electrons)
```

Used in: All Recipes

Defined in: SpirouDRS.spirouConfig.spirouKeywords

- **Gain (`kw_GAIN`)**

The gain [e-/ADU]

```
kw_GAIN = ["GAIN", "0.0", "gain (electrons/ADU)"]
```

HEADER file entry:

```
GAIN       = 0.0 \ gain (electrons/ADU)
```

Used in: All Recipes

Defined in: SpirouDRS.spirouConfig.spirouKeywords

- **Exposure time (`kw_EXPTIME`)**

The integration time in seconds

```
kw_EXPTIME = ["EXPTIME", "0.0", "Integration time (seconds)"]
```

HEADER file entry:

```
EXPTIME    = 0.0 \ Integration time (seconds)
```

Used in: All Recipes

Defined in: SpirouDRS.spirouConfig.spirouKeywords

Chapter 7

User modifiable 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.

7.1 Variable file locations

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 3.8 and are located in `DRS_ROOT /config /../config /config.txt`.

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

7.2 Global variables

- 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`

- **Debug mode (DRS_DEBUG)**

Defines whether we should run the DRS in debug mode. Certain print/log statements and certain graphs only plot in debug mode. On an error the option to enter the python debugger is asked (allows user to look into functions/current memory and see what variables are currently defined. Value must be an integer. Value must be an integer where:

- 0 = No debug
- 1 = basic debugging on errors (prompted to enter python debugger)
- 2 = Same as 1 and recipes specific (plots and some code runs)

DRS_DEBUG = 0

Used in: All Recipes

Defined in: ../config /config.txt

- **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

7.3 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

- **The installation directory (DRS_ROOT)**

Defines the installation directory (**DRS_ROOT**). Value must be a string containing a valid file location.

DRS_ROOT = /drs/INTROOT/

Used in: All Recipes

Defined in: ../config /config.txt

- 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

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

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_REDUCE = /drs/data/reduced

Used in: All Recipes

Defined in: ../config /config.txt

- 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

- 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

- 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

7.4 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
```

- 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
```

- 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
```

- Available fiber types (**fiber_types**)

Defines the type of fiber we have (used in various codes). These 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`

7.5 Fiber variables

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

- **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

- **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

- **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

- **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

- **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`

- **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`

- **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`

- **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_spirou` both 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`

- **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 `ic_calibDB_filename` used will be `LOC_{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`

- **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 `ic_calibDB_filename` used will be `ORDER_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`

- **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`

7.6 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`

- 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`

- 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`

- 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 (not equal to) the value of 'histo_range_high'

histo_range_low = -0.5

Used in: `cal_DARK_spirou`

Defined in: `constants_SPIROU.txt`

- 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`

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

Defines the bad pixel cut limit in ADU/s.

$$badpixels = (image > dark_cut_limit) \text{ OR (non-finite)} \quad (7.1)$$

dark_cutlimit = 100.0

Used in: [cal_DARK_spirou](#)

Defined in: [constants_SPIROU.txt](#)

7.7 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`

- **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`

- **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`

- **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`

- **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

- **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:

$$(\text{top of the row} < \text{ic_image_gap}) \text{ OR } (\text{bottom of the row} > \text{ic_image_gap}) \quad (7.2)$$

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

- **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

- **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

- **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`

- **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`

- **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`

- **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`

- 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`

- 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`

- 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:

$$\max(RMS) > \text{ic_max_rms_center} \quad (7.3)$$

ic_max_rms_center = 0.2

Used in: `cal_loc_RAW_spirou`

Defined in: `constants_SPIROU.txt`

- **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:

$$\max(|\text{residuals}|) > \text{ic_max_ptp_center} \quad (7.4)$$

`ic_max_ptp_center` = 0.2

Used in: `cal_loc_RAW_spirou`
 Defined in: `constants_SPIROU.txt`

- **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:

$$\max(|\text{residuals}|)/\text{RMS} > \text{ic_ptporms_center} \quad (7.5)$$

`ic_ptporms_center` = 8.0

Used in: `cal_loc_RAW_spirou`
 Defined in: `constants_SPIROU.txt`

- **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:

$$\max(RMS) > \text{ic_max_rms_width} \quad (7.6)$$

`ic_max_rms_fwhm` = 1.0

Used in: `cal_loc_RAW_spirou`
 Defined in: `constants_SPIROU.txt`

- **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:

$$\max(|\text{residuals}/\text{data}|) \times 100 > \text{ic_max_ptp_fracfwhm} \quad (7.7)$$

`ic_max_ptp_fracfwhm` = 1.0

Used in: `cal_loc_RAW_spirou`
 Defined in: `constants_SPIROU.txt`

- **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`

- **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`

7.8 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`

- **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 $\pm \times 2$ 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`

- **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` = 4

Used in: `cal_SLIT_spirou`

Defined in: `constants_SPIROU.txt`

- **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`

7.9 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`

- 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 $2 \times \text{ic_bkgr_window}$ 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`

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

Defines the number of orders in the tilt measurement file (TILT key in the `ic_calibDB_filename`). 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`

- 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`

- **Half size blaze window (`ic_extfbblaz`)**

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_extfbblaz` = 50

Used in: `cal_FF_RAW_spirou`
 Defined in: `constants_SPIROU.txt`

- **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`

- **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`

- **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`

7.10 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`

- **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`

- **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_spirouAB` or `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`

- **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`

- **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`

7.11 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.

$$snr = flux / \sqrt{(flux + noise^2)} \quad (7.8)$$

Value must be a float larger than zero.

ic_drift_noise = 100.0

Used in: cal_DRIFT_RAW_spirou
Defined in: constants_SPIROU.txt

- 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

- 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 an integer larger than zero.

ic_drift_boxsize = 12

Used in: cal_DRIFT_RAW_spirou
Defined in: constants_SPIROU.txt

- 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

- 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`

- 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`

- 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`

- 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`

- **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**

- **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 7.9) or 'median' (Equation 7.10) for **cal_DRIFT_RAW_spirou**.

$$\text{drift} = \frac{\sum (\text{drift}_i * w_i)}{\sum w_i} \quad (7.9)$$

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

$$\text{drift} = \text{median}(\text{drift}_i) \quad (7.10)$$

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**

- **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 7.11) or 'median' (Equation 7.12) for **cal_DRIFT_E2DS_spirou**.

$$\text{drift} = \frac{\sum (\text{drift}_i * w_i)}{\sum w_i} \quad (7.11)$$

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

$$\text{drift} = \text{median}(\text{drift}_i) \quad (7.12)$$

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**

- **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`

7.12 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**

- 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**

- 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**

- 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 \pm 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**

- **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`

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

Defines the half-box size (i.e. central position \pm 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`

- **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 equivalent 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`

- **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`

- 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:

$$\text{normalized FP FWHM} > \text{drift_peak_norm_width_cut} \quad (7.13)$$

this is equivalent to:

$$\text{FP FWHM} > (\text{drift_peak_exp_width} + \text{drift_peak_norm_width_cut}) \quad (7.14)$$

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

$$\text{drift_peak_exp_width} = 0.8$$

Used in: `cal_DRIFT-PEAK_E2DS_spirou`
 Defined in: `constants_SPIROU.txt`

- 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:

$$\text{normalized FP FWHM} > \text{drift_peak_norm_width_cut} \quad (7.15)$$

this is equivalent to:

$$\text{FP FWHM} > (\text{drift_peak_exp_width} + \text{drift_peak_norm_width_cut}) \quad (7.16)$$

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.

$$\text{drift_peak_norm_width_cut} = 0.2$$

Used in: `cal_DRIFT-PEAK_E2DS_spirou`
 Defined in: `constants_SPIROU.txt`

- 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.

$$\text{drift_peak_getdrift_gaussfit} = \text{False}$$

Used in: `cal_DRIFT-PEAK_E2DS_spirou`
 Defined in: `constants_SPIROU.txt`

- **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:

$$\text{coefficient}_{\text{order}} > \text{drift_peak_pearsonr_cut} \quad (7.17)$$

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`

- **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`

- **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`

- **Selected order for linelist vs log Amplitude plot (`drift_peak_selected_order`)**

Defines the selected order to plot the wave vs extracted spectrum for overplotting on the line list against log amplitude plot. Value must be an integer between 0 and the number of orders

`drift_peak_selected_order` = 30

Used in: `cal_DRIFT-PEAK_E2DS_spirou`
 Defined in: `constants_SPIROU.txt`

7.13 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`

- 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`

- 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`

- 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`

- **Bad pixel normalisation percentile (`badpix_norm_percentile`)**

Defines the percentile at which the bad pixels are normalised to in order to locate bad and dead pixels.

`badpix_norm_percentile` = 90.0

Used in: `cal_BADPIX_spirou`

Defined in: `constants_SPIROU.txt`

7.14 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:

$$\text{Median Flux} < \text{qc_max_darklevel} \quad (7.18)$$

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

$$\text{qc_max_darklevel} = 0.5$$

Used in: `cal_DARK_spirou`
 Defined in: `constants_SPIROU.txt`

- 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:

$$\text{dead pixels} = (\text{pixel value} > \text{dark_cutlimit}) \text{ and } (\text{pixel value} \neq \text{NaN}) \quad (7.19)$$

$$\text{Percentage of dead pixels} < \text{qc_max_dead} \quad (7.20)$$

$$\text{qc_max_dead} = 20.0$$

Used in: `cal_DARK_spirou`
 Defined in: `constants_SPIROU.txt`

- 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:

$$\text{bad dark pixels} = \text{pixel value} > \text{dark_cutlimit} \quad (7.21)$$

$$\text{Percentage of bad dark pixels} < \text{qc_max_dark} \quad (7.22)$$

$$\text{qc_max_dark} = 6.0$$

Used in: `cal_DARK_spirou`
 Defined in: `constants_SPIROU.txt`

- **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`

- **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:

$$\text{Number of rejected orders in center fit} > \text{qc_loc_maxlocfit_removed_ctr} \quad (7.23)$$

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`

- **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:

$$\text{Number of rejected orders in width fit} > \text{qc_loc_maxlocfit_removed_width} \quad (7.24)$$

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`

- **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:

$$\text{Mean rms center fit} > \text{qc_loc_rmsmax_center} \quad (7.25)$$

Value must be a float greater than zero.

`qc_loc_rmsmax_center` = 100

Used in: `cal_loc_RAW_spirou`

Defined in: `constants_SPIROU.txt`

- **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:

$$\text{Mean rms width fit} > \text{qc_loc_rmsmax_fwhm} \quad (7.26)$$

Value must be a float greater than zero.

`qc_loc_rmsmax_fwhm` = 500

Used in: `cal_loc_RAW_spirou`

Defined in: `constants_SPIROU.txt`

- **Maximum allowed RMS (`qc_ff_rms`)**

Defines the maximum RMS allowed to accept a flat-field for calibration. Value must be a float greater than zero.

`qc_ff_rms` = 0.12

Used in: `cal_FF_RAW_spirou`

Defined in: `constants_SPIROU.txt`

- **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`

- **Maximum RMS allowed for slit TILT (`qc_slit_rms`)**

Defines the maximum allowed RMS in the calculated TILT to add TILT profile to the `calibration database`. Value must be a float larger than zero.

`qc_slit_rms` = 0.1

Used in: `cal_SLIT_spirou`

Defined in: `constants_SPIROU.txt`

- Minimum angle allowed for slit TILT (**qc_slit_min**)

Defines the minimum tilt angle allowed to add TILT profile to the calibration database. Value must be a float and must be less than **qc_slit_max**

qc_slit_min = -8.0

Used in: **cal_SLIT_spirou**

Defined in: **constants_SPIROU.txt**

- Maximum angle allowed for slit TILT (**qc_slit_max**)

Defines the maximum tilt angle allowed to add TILT profile to the calibration database. Value must be a float and must be greater than **qc_slit_min**

qc_slit_max = 0.0

Used in: **cal_SLIT_spirou**

Defined in: **constants_SPIROU.txt**

- 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_extract_RAW_spirou**

Defined in: **constants_SPIROU.txt**

7.15 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

- 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

- Calibration database duplicate key handler (**calib_db_match**)

Defines the mechanism to use in deciding between duplicate keys in the calibration database file. Value must be a string and must be either 'older' or 'closest'. If 'older' the calibration database will only use keys that are older than the timestamp in the input fits file (first argument) using the key kw_ACQTIME_KEY

calib_db_match = 'closest'

Used in: All Recipes

Defined in: constants_SPIROU.txt

7.16 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.

`PRINT_LEVEL` = all

Used in: All Recipes
Defined in: ../config /config.txt

- **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.

`LOG_LEVEL` = all

Used in: All Recipes
Defined in: ../config /config.txt

- **Toggle coloured log (`COLOURED_LOG`)**

Defines whether the log (printed to the standard output) is coloured . Value must be True or 1 to colour the log or False or 0 to use the default console colour throughout.

`COLOURED_LOG` = True

Used in: All Recipes
Defined in: ../config /config.txt

Chapter 8

The Recipes

8.1 The cal_DARK recipe

Dark with short exposure time (5min, to be defined during AT-4) to check if read-out noise, dark current and hot pixel mask are consistent with the ones obtained during technical night. Quality control is done automatically by the pipeline

8.1.1 The inputs

The input of `cal_DARK_spirou` is as follows:

```
>> cal_DARK_spirou.py night_repository filenames
```

for example:

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

or

```
Python/Ipynon
import cal_DARK_spirou
night_repository = '20170710'
filenames = ['dark_dark02d406.fits']
cal_DARK_spirou.main(night_repository, file=filenames)
```

where 'night_repository' defines arg_night_name and 'filenames' define the list of files in arg_file_names. All files in filenames must be valid python strings separated by a space (command line) or in a line (python).

Filename prefixes allowed are:

- dark_dark

8.1.2 The outputs

The outputs of `cal_DARK_spirou` are as follows:

- darkfile in form:

```
{reduced_dir}/{date prefix}_{file}.fits
```

- darkbadpixfile in form:

```
{reduced_dir}/{date prefix}_{file}_badpixel.fits
```

where 'date prefix' is constructed from arg_night_name and the file name is the first file in arg_file_names.

For example for `reduced_dir='/drs/data/reduced/20170710'` and `arg_file_names=['dark_dark02d406.fits']` the output files would be:

- `/drs/data/reduced/20170710/20170710_dark_dark02d406.fits`
- `/drs/data/reduced/20170710/20170710_dark_dark02d406_badpixel.fits`

8.1.3 Summary of procedure

1. adds defined 'dark_dark' files together
2. resizes the image
3. calculates the fraction of dead pixels [full, blue part, red part]
4. calculates median dark level [full, blue part, red part]
5. calculates threshold of dark level to retain
6. removes dead pixels by setting them to 0
7. does some quality control
8. updates calibDB with key "DARK"

8.1.4 Quality Control

There are currently three quality control checks for `cal_DARK_spirou`

- Unexpected median dark level if:

$$\text{Median Flux} > \text{qc_max_darklevel} \quad (8.1)$$

- Unexpected fraction of dead pixels if:

$$\text{Number of dead pixels} > \text{qc_max_dead} \quad (8.2)$$

- Unexpected fraction of dark pixels if:

$$\text{Number of bad dark pixels} > \text{qc_max_dark} \quad (8.3)$$

If none of these quality control criteria are valid then the output file is passed into the [calibration database](#) with key 'DARK' for the 'darkfile' and 'BADPIX' for the 'darkbadpixfile'.

For example the following lines are added to the [calibration database](#) for `arg_night_name = "20170710"` and `arg_file_names = "dark_dark02d406.fits"` .

In calibration database file

```
DARK 20170710 20170710_dark_dark02d406.fits 2017-07-10-12:37:48.260000 1499690268.26
BADPIX 20170710 20170710_dark_dark02d406_badpixel.fits 2017-07-10-12:37:48.260000 1499690268.26
```

8.1.5 Example working run

An example run where everything worked is below:

example

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

```
HH:MM:SS.S - || *****
HH:MM:SS.S - || * SPIROU @(#) Geneva Observatory (VERSION)
HH:MM:SS.S - || *****
HH:MM:SS.S - ||(dir_data_raw)      DRS_DATA_RAW=/drs/data/raw
HH:MM:SS.S - ||(dir_data_reduc)    DRS_DATA_REduc=/drs/data/reduced
HH:MM:SS.S - ||(dir_calib_db)     DRS_CALIB_DB=/drs/data/calibDB
HH:MM:SS.S - ||(dir_data_msg)    DRS_DATA_MSG=/drs/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=/drs/data/tmp/
HH:MM:SS.S - ||
HH:MM:SS.S - ||      DRS_INTERACTIVE is not set, running on-line mode
HH:MM:SS.S - ||      DRS_DEBUG is set, debug mode level:1
HH:MM:SS.S - |ipython:2d406|Now running : ipython on file(s): dark_dark02d406.fits
HH:MM:SS.S - |ipython:2d406|On directory /drs/data/raw/20170710
HH:MM:SS.S - |ipython:2d406|ICDP_NAME loaded from: /drs/INTROOT/config/constants_SPIROU.py
HH:MM:SS.S - * |ipython:2d406|Correct type of image for dark (dark_dark)
HH:MM:SS.S - * |ipython:2d406|Now processing Image TYPE UNKNOWN with ipython recipe
HH:MM:SS.S - |ipython:2d406|Reading Image /drs/data/raw/20170710/dark_dark02d406.fits
HH:MM:SS.S - |ipython:2d406|Image 2048 x 2048 loaded
HH:MM:SS.S - * |ipython:2d406|Dark Time = 597.489 s
HH:MM:SS.S - |ipython:2d406|Doing Dark measurement
HH:MM:SS.S - * |ipython:2d406|In Whole det: Frac dead pixels= 14.7 % - Median= 0.35 ADU/s - Percent
HH:MM:SS.S - | [5:95]= 0.08-99.57 ADU/s
HH:MM:SS.S - * |ipython:2d406|In Blue part: Frac dead pixels= 1.0 % - Median= 0.15 ADU/s - Percent
HH:MM:SS.S - | [5:95]= 0.09-0.53 ADU/s
HH:MM:SS.S - * |ipython:2d406|In Red part : Frac dead pixels= 20.5 % - Median= 2.11 ADU/s - Percent
HH:MM:SS.S - | [5:95]= 0.18-232.09 ADU/s
HH:MM:SS.S - * |ipython:2d406|Frac pixels with DARK > 100.0 ADU/s = 4.3 %
HH:MM:SS.S - @ |python warning|Line 138 warning reads: invalid value encountered in greater
HH:MM:SS.S - * |ipython:2d406|Total Frac dead pixels (N.A.N) + DARK > 100.0 ADU/s = 18.9 %
HH:MM:SS.S - * |ipython:2d406|QUALITY CONTROL SUCCESSFUL - Well Done -
HH:MM:SS.S - |ipython:2d406|Saving Dark frame in 20170710_dark_dark02d406.fits
HH:MM:SS.S - @ |python warning|Line 980 warning reads: Card is too long, comment will be truncated.
HH:MM:SS.S - |ipython:2d406|Saving Bad Pixel Map in 20170710_dark_dark02d406_badpixel.fits
HH:MM:SS.S - @ |python warning|Line 980 warning reads: Card is too long, comment will be truncated.
HH:MM:SS.S - * |ipython:2d406|Updating Calib Data Base with DARK
HH:MM:SS.S - * |ipython:2d406|Updating Calib Data Base with BADPIX
HH:MM:SS.S - * |ipython:2d406|Recipe ipython has been succesfully completed
```

8.1.6 Interactive mode

In interactive mode (`DRS_PLOT = 1`) three figures will also appear (see Figure 8.1).

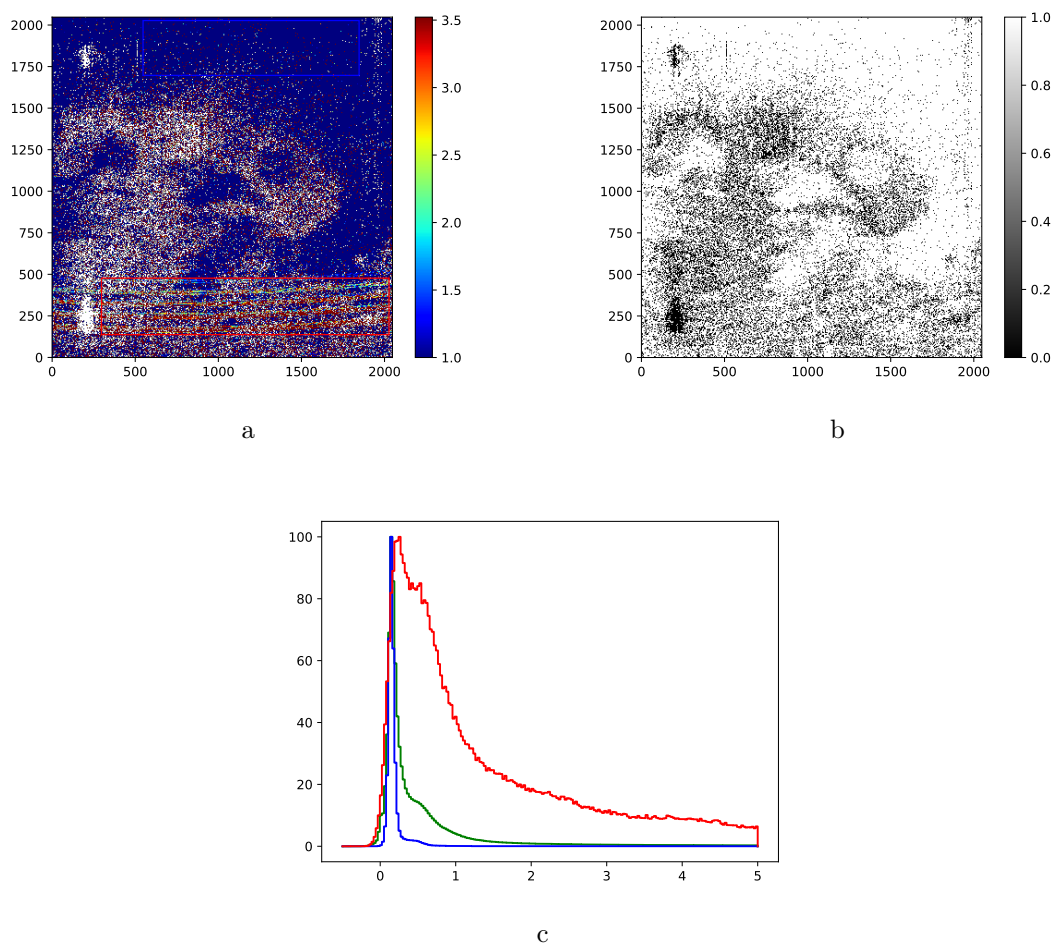


Figure 8.1: **(a)** The image with over-plot red and blue regions (red/blue rectangles). **(b)** The bad pixel mask, bad pixels have a value=1 (in black) and good pixels have a value=0 (in white). **(c)** Histograms of the image regions, the full image (in green), the blue section (in blue) and the red section (in red).

8.2 The cal_BADPIX recipe

Recipe to generate the bad pixel map.

8.2.1 The inputs

The input of `cal_BADPIX_spirou` is as follows:

```
>> cal_BADPIX_spirou.py night_repository flatfile darkfile
```

for example:

example

```
>> cal_BADPIX_spirou.py 20170710 flat_flat02f10.fits dark_dark02d406.fits
```

or

Python/Ipypthon

```
import cal_DARK_spirou
night_repository = '20170710'
darkfile = 'dark_dark02d406.fits'
flatfile = 'flat_flat02f10.fits'
cal_DARK_spirou.main(night_repository, flatfile=flatfile, darkfile=darkfile)
```

where 'night_repository' defines `arg_night_name` and 'filenames' define the list of files in `arg_file_names`. All files in filenames must be valid python strings separated by a space (command line) or in a line (python) and must have the following prefixes: File prefixes allowed:

- flat_flat (flatfile)
- dark_dark (darkfile)

8.2.2 The outputs

The outputs of `badpixelfits` are as follows:

- `badpixelfits` in form:

```
{reduced_dir}/{date prefix}_{file}_badpixelfits.fits
```

where 'date prefix' is constructed from `arg_night_name` and the file name is the flatfile name.

for example for `reduced_dir='/drs/data/reduced/20170710'` and `flatfile='flat_flat02f10.fits'` the output file would be:

- `/drs/data/reduced/20170710/20170710_flat_flat02f10_badpixelfits.fits`

8.2.3 Summary of procedure

1. Normalise the flats
2. Look for isolated hot pixels
3. Calculate how much pixels deviate compared to expected values

4. Select hot pixels compared to neighbours
5. Combine bad pixel map
6. Save bad pixel mask to file

8.2.4 Quality Control

There are no quality control parameters for `cal_BADPIX_spirou`.

The output file is passed into the `calibration database` with key 'BADPIX' for the 'badpixfile'.

For example the following lines are added to the `calibration database` for `arg_night_name = "20170710"` ,
`flatfile = "flat_flat02f10.fits"` and `darkfile = "dark_dark02d406.fits"` .

In calibration database file

```
BADPIX 20170710 20170710_flat_flat02f10_badpixel.fits 2017-07-10-13:07:49.470000 1499692069.47
```


8.2.5 Example working run

An example run where everything worked is below:

example

```
>> cal_BADPIX_spirou.py 20170710 flat_flat02f10.fits dark_dark02d406.fits
```

```
HH:MM:SS.S - || *****
HH:MM:SS.S - || * SPIROU @(#) Geneva Observatory (VERSION)
HH:MM:SS.S - || *****
HH:MM:SS.S - ||(dir_data_raw)      DRS_DATA_RAW=/drs/data/raw
HH:MM:SS.S - ||(dir_data_reduc)     DRS_DATA_REduc=/drs/data/reduced
HH:MM:SS.S - ||(dir_calib_db)      DRS_CALIB_DB=/drs/data/calibDB
HH:MM:SS.S - ||(dir_data_msg)     DRS_DATA_MSG=/drs/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=/drs/data/tmp
HH:MM:SS.S - ||
HH:MM:SS.S - ||      DRS_INTERACTIVE is not set, running on-line mode
HH:MM:SS.S - ||      DRS_DEBUG is set, debug mode level:1
HH:MM:SS.S - |cal_BADPIX_spirou|Now running : cal_BADPIX_spirou with:
HH:MM:SS.S - |cal_BADPIX_spirou|      -- flatfile=flat_flat02f10.fits
HH:MM:SS.S - |cal_BADPIX_spirou|      -- darkfile=dark_dark02d406.fits
HH:MM:SS.S - |cal_BADPIX_spirou|Config Error: ICDP_NAME loaded from: /drs/spirou_py3/INTROOT/config/
              constants_SPIROU.py
HH:MM:SS.S - * |cal_BADPIX_spirou|Now processing Image TYPE FLAT with cal_BADPIX_spirou recipe
HH:MM:SS.S - * |cal_BADPIX_spirou|Now processing Image TYPE DARK with cal_BADPIX_spirou recipe
HH:MM:SS.S - |cal_BADPIX_spirou|Reading FLAT Image /drs/data/raw/20170710/flat_flat02f10.fits
HH:MM:SS.S - |cal_BADPIX_spirou|FLAT Image 2048 x 2048 loaded
HH:MM:SS.S - |cal_BADPIX_spirou|Reading DARK Image /drs/data/raw/20170710/dark_dark02d406.fits
HH:MM:SS.S - |cal_BADPIX_spirou|DARK Image 2048 x 2048 loaded
HH:MM:SS.S - |cal_BADPIX_spirou|Normalising the flat
HH:MM:SS.S - |cal_BADPIX_spirou|Looking for bad pixels
HH:MM:SS.S - |cal_BADPIX_spirou|Fraction of hot pixels from dark: 3.01 %
HH:MM:SS.S - |cal_BADPIX_spirou|Fraction of bad pixels from flat: 1.66 %
HH:MM:SS.S - |cal_BADPIX_spirou|Fraction of non-finite pixels in dark: 20.76 %
HH:MM:SS.S - |cal_BADPIX_spirou|Fraction of non-finite pixels in flat: 14.66 %
HH:MM:SS.S - |cal_BADPIX_spirou|Fraction of bad pixels with all criteria: 24.87 %
HH:MM:SS.S - * |cal_BADPIX_spirou|QUALITY CONTROL SUCCESSFUL - Well Done -
HH:MM:SS.S - |cal_BADPIX_spirou|Saving Bad Pixel Map in 20170710_flat_flat02f10_badpixel.fits
HH:MM:SS.S - @ |python warning Line 980  warning reads: Card is too long, comment will be truncated.|
HH:MM:SS.S - * |cal_BADPIX_spirou|Updating Calib Data Base with BADPIX
HH:MM:SS.S - * |cal_BADPIX_spirou|Recipe cal_BADPIX_spirou has been successfully completed
```

8.3 The cal_loc recipe

Locates the orders on the ‘dark_flat’ or ‘flat_dark’ images.

8.3.1 The inputs

The input of `cal_loc_RAW_spirou` is as follows:

```
>> cal_loc_RAW_spirou.py night_repository filenames
```

for example:

example

```
>> cal_loc_RAW_spirou.py 20170710 flat_dark02f10.fits
```

or

Python/Ipypthon

```
import cal_loc_RAW_spirou
night_repository = '20170710'
filenames = ['flat_dark02f10.fits']
cal_loc_RAW_spirou.main(night_repository, files=filenames)
```

where ‘night_repository’ defines `arg_night_name` and ‘filenames’ define the list of files in `arg_file_names`. All files in `filenames` must be valid python strings separated by a space (command line) or in a line (python) and must have the following prefixes: File prefixes allowed:

- dark_flat
- flat_dark

8.3.2 The outputs

The outputs of `cal_loc_RAW_spirou` are as follows:

- order_profile in form:

```
{reduced_dir}/{date prefix}_{file}_order_profile_{fiber}.fits
```

- locofitsfile in form:

```
{reduced_dir}/{date prefix}_{file}_loco_{fiber}.fits
```

- locofitsfile2 in form:

```
{reduced_dir}/{date prefix}_{file}_fwhm-order_{fiber}.fits
```

- locofitsfile3 in form:

```
{reduced_dir}/{date prefix}_{file}_with-order_{fiber}.fits
```

where ‘date prefix’ is constructed from `arg_night_name` and the file name is the first file in `arg_file_names`.

For example for `reduced_dir='/drs/data/reduced/20170710'` and `arg_file_names=['flat_dark02f10.fits']` the output files would be:

- `/drs/data/reduced/20170710/20170710_flat_dark02f10_order_profile_{fiber}.fits`
- `/drs/data/reduced/20170710/20170710_flat_dark02f10_loco_{fiber}.fits`
- `/drs/data/reduced/20170710/20170710_flat_dark02f10_fwhm-order_{fiber}.fits`
- `/drs/data/reduced/20170710/20170710_flat_dark02f10_with-order_{fiber}.fits`

8.3.3 Summary of procedure

1. adds all defined 'dark_flat' or 'flat_dark' files together
2. corrects for darks
3. resizes the image
4. constructs 'order_profile' image
5. locates the central pixel of each order
6. steps out in large steps along the order (toward beginning and end)
7. fits the position of each order (using a small 2D box around each fit point)
 - includes a rejection of bad points (while loop)
8. fits the width of each order (using a small 2D box around each fit point)
 - includes a rejection of bad points (while loop)
9. saves the 'order_profile' image (with a superposition of the fit orders as zero values)
10. does some quality control
11. updates calibDB with key "LOC_fiber" where fiber = [AB, C] etc

8.3.4 Quality Control

There are currently five quality control checks for `cal_loc_RAW_spirou`

- Too many rejected orders in center position fit:

$$\text{Number of rejected orders in center fit} > \text{qc_loc_maxlocfit_removed_ctr} \quad (8.4)$$

- Too many rejected orders in width fit:

$$\text{Number of rejected orders in width fit} > \text{qc_loc_maxlocfit_removed_wid} \quad (8.5)$$

- RMS on center fit too high:

$$\text{Mean rms center fit} > \text{qc_loc_rmsmax_center} \quad (8.6)$$

- RMS on width fit too high:

$$\text{Mean rms width fit} > \text{qc_loc_rmsmax_fwhm} \quad (8.7)$$

- Abnormal number of identified orders:

$$\text{Number of orders found} \neq \text{qc_loc_nbo} \quad (8.8)$$

If none of these quality control criteria are valid then the output file is passed into the [calibration database](#) with key 'LOC_{fiber}' for the 'locofitsname' file.

For example the following lines are added to the [calibration database](#) for `arg_night_name = "20170710"` and `arg_file_names = ["flat_dark02f10.fits"]`.

In calibration database file

```
LOC_AB 20170710 20170710_flat_dark02f10_loco_AB.fits 2017-07-10-13:07:08.460000 1499692028.46
```

8.3.5 Example working run

An example run where everything worked is below:

example

```
>> cal_loc_RAW_spirou.py 20170710 flat_dark02f10.fits
```

```
HH:MM:SS.S - || *****
HH:MM:SS.S - || * SPIROU @(#) Geneva Observatory (0.1.015)
HH:MM:SS.S - || *****
HH:MM:SS.S - ||(dir_data_raw)      DRS_DATA_RAW=/drs/data/raw
HH:MM:SS.S - ||(dir_data_reduc)    DRS_DATA_REDUCE=/drs/data/reduced
HH:MM:SS.S - ||(dir_calib_db)     DRS_CALIB_DB=/drs/data/calibDB
HH:MM:SS.S - ||(dir_data_msg)    DRS_DATA_MSG=/drs/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=/drs/data/tmp
HH:MM:SS.S - ||                  DRS_INTERACTIVE is not set, running on-line mode
HH:MM:SS.S - ||                  DRS_DEBUG is set, debug mode level:1
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|Now running : cal_loc_RAW_spirou on file(s): flat_dark02f10.
fits
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|On directory /drs/data/raw/20170710
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|ICDP_NAME loaded from: /drs/spirou_py3/INTROOT/config/
constants_SPIROU.py
HH:MM:SS.S - * |cal_loc_RAW_spirou:02f10|Correct type of image for localisation (dark_flat or
flat_dark)
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|Calibration file: 20170710_flat_flat02f10_badpixel.fits
already exists - not copied
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|Calibration file: 20170710_dark_dark02d406.fits already
exists - not copied
...
```

```

HH:MM:SS.S - * |cal_loc_RAW_spirou:02f10|Now processing Image TYPE UNKNOWN with cal_loc_RAW_spirou
recipe
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|Reading Image /drs/data/raw/20170710/flat_dark02f10.fits
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|Image 2048 x 2048 loaded
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|Doing Dark Correction using /drs/data/calibDB/20170710
_dark_dark02d406.fits
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|Image format changed to 1930x2035
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|Saving processed raw frame in 20170710
_flat_dark02f10_order_profile_AB.fits
HH:MM:SS.S - @ |python warning Line 980 warning reads: Card is too long, comment will be truncated.|
HH:MM:SS.S - * |cal_loc_RAW_spirou:02f10|Updating Calib Data Base with ORDER_PROFILE_AB
HH:MM:SS.S - * |cal_loc_RAW_spirou:02f10|Maximum flux/pixel in the spectrum: 82388.4 [e-]
HH:MM:SS.S - * |cal_loc_RAW_spirou:02f10|Average background level: 1.39 [%]
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|Searching order center on central column
HH:MM:SS.S - * |cal_loc_RAW_spirou:02f10|On fiber AB 36 orders have been detected on 2 fiber(s)
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|ORDER: 0 center at pixel 102.5 width 11.6 rms 0.049
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - center fit rms/ptp/sigrms: 0.049/0.140/2.855 with 0
rejected points
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - width fit rms/ptp/ptp%: 0.434/0.740/6.726 with 0 rejected
points
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|ORDER: 1 center at pixel 116.9 width 11.4 rms 0.073
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - center fit rms/ptp/sigrms: 0.073/0.164/2.240 with 0
rejected points
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - width fit rms/ptp/ptp%: 0.448/0.794/6.618 with 0 rejected
points
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|ORDER: 2 center at pixel 149.8 width 11.7 rms 0.049
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - center fit rms/ptp/sigrms: 0.049/0.116/2.374 with 0
rejected points
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - width fit rms/ptp/ptp%: 0.421/0.881/8.006 with 0 rejected
points
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|ORDER: 3 center at pixel 164.2 width 11.5 rms 0.075
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - center fit rms/ptp/sigrms: 0.075/0.173/2.309 with 0
rejected points
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - width fit rms/ptp/ptp%: 0.420/0.806/7.119 with 0 rejected
points
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|ORDER: 4 center at pixel 196.6 width 11.5 rms 0.051
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - center fit rms/ptp/sigrms: 0.051/0.140/2.762 with 0
rejected points
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - width fit rms/ptp/ptp%: 0.403/0.989/8.889 with 0 rejected
points
...
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|ORDER: 69 center at pixel 1801.4 width 10.4 rms 0.168
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| center fit converging with rms/ptp/sigrms:
0.168/1.465/8.710
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - center fit rms/ptp/sigrms: 0.067/0.167/2.490 with 1
rejected points
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| fwhm fit converging with rms/ptp/ptp%:
0.535/2.337/29.218
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| fwhm fit converging with rms/ptp/ptp%:
0.478/1.105/11.053
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - width fit rms/ptp/ptp%: 0.466/0.936/9.357 with 2 rejected
points
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|ORDER: 70 center at pixel 1851.8 width 7.1 rms 2.565

```

```

HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|      center fit converging with rms/ptp/sigrms:
2.565/8.719/3.399
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - center fit rms/ptp/sigrms: 0.077/0.158/2.047 with 22
rejected points
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|      fwhm fit converging with rms/ptp/ptp%:
1.070/2.674/44.574
...
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - width  fit rms/ptp/ptp%: 0.300/0.623/9.965 with 17
rejected points
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|ORDER: 71 center at pixel 1869.1 width 10.7 rms 0.296
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|      center fit converging with rms/ptp/sigrms:
0.296/0.889/3.008
...
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - center fit rms/ptp/sigrms: 0.099/0.199/1.998 with 30
rejected points
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|      fwhm fit converging with rms/ptp/ptp%:
0.995/3.544/23.627
...
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10| - width  fit rms/ptp/ptp%: 0.468/0.838/8.383 with 23
rejected points
HH:MM:SS.S - * |cal_loc_RAW_spirou:02f10|On fiber AB 72 orders geometry have been measured
HH:MM:SS.S - * |cal_loc_RAW_spirou:02f10|Average uncertainty on position: 66.42 [mpix]
HH:MM:SS.S - * |cal_loc_RAW_spirou:02f10|Average uncertainty on width: 381.93 [mpix]
HH:MM:SS.S - * |cal_loc_RAW_spirou:02f10|QUALITY CONTROL SUCCESSFUL - Well Done -
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|Saving localization information in file: 20170710
_flat_dark02f10_loco_AB.fits
HH:MM:SS.S - @ |python warning Line 980  warning reads: Card is too long, comment will be truncated.|
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|Saving FWHM information in file: 20170710_flat_dark02f10_fwhm
-order_AB.fits
HH:MM:SS.S - @ |python warning Line 980  warning reads: Card is too long, comment will be truncated.|
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|Saving localization image with superposition of orders in
HH:MM:SS.S - |cal_loc_RAW_spirou:02f10|file: 20170710_flat_dark02f10_with-order_AB.fits
HH:MM:SS.S - * |cal_loc_RAW_spirou:02f10|Updating Calib Data Base with LOC_AB
HH:MM:SS.S - * |cal_loc_RAW_spirou:02f10|Recipe cal_loc_RAW_spirou has been successfully completed

```

8.4 The cal_SLIT recipe

8.5 The cal_FF recipe

8.6 The cal_extract recipes

8.7 The cal_DRIFT recipes

8.8 The cal_HC recipe

8.9 The `cal_WAVE` recipe

8.10 The cal_CCF recipe

8.11 The `pol_spirou` recipe

8.12 The validation recipes recipe