SPIRou Data Reduction Software

Developer Guide

0.0.1

For DRS SPIRou0.0.1

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

2017-11-27

Abstract

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

Contents

Chapter 1

Introduction

1.1 Code blocks

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

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

text

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

These can also be shell scripts in a certain language:

bash

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

tcsh/csh

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

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

CMD input

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

The following denotes a command line print out

Command line output

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

The following denotes a python terminal or python script

Python/Ipython

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

Chapter 2

Installation

2.1 Introduction

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

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

2.2 Download

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

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

CMD input

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

- use SVN:

CMD input

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

- use ssh:

CMD input

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

2.3 Prerequisites

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

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

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

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

2.3.1 Anaconda python distribution

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

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

2.3.2 Separate python installation

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

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

2.4 Installation Linux and macOS

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

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

2.4.1 Extraction

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

CMD input

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

2.4.2 Modify environmental settings

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

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

bash

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

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

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

tcsh/csh

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

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

2.4.3 Make recipes executable

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

CMD input

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

2.5 Installation Windows

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

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

2.5.1 How to modify environmental settings in windows

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

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

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

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

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

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

2.6 Setting up the DRS

Before running the DRS one must set the data paths.

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

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

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

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

The following keywords can be changed:

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

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

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

2.7 Validating Installation on Linux and macOS

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

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

- To validate running from command line type:

CMD input

```
>> cal_validate_spirou.py
```

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

CMD input

```
>> python cal_validate_spirou.py
>> ipython cal_validate_spirou.py
```

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

Python/Ipython

```
run cal_validate_spirou.py
```

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

Python/Ipython

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

If validation is successful the following should appear:

Command line output

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

2.8 Validating Installation on Windows

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

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

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

CMD input

```
>> python cal_validate_spirou.py
>> ipython cal_validate_spirou.py
```

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

Python/Ipython

```
run cal_validate_spirou.py
```

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

Python/Ipython

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

If validation is successful the following should appear:

Command line output

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

Chapter 3

Data Architecture

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

3.1 Installed file structure

The file structure should look as follows:

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

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

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

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

3.2 The Installation root directory

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

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

3.2.1 The bin directory

The bin directory is located in the {INSTALL_ROOT} directory. This contains all the recipes that can be used. A detailed description of all recipes can be found in Chapter 9 but are listed here for completeness.

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

3.2.2 The SPIROU module directory

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

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

The modules are described in detail in Chapter 10.

3.3 The data root directory

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

3.3.1 The raw and reduced data directories

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

The file structure can be seen below:

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

3.4 The calibration database directory

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

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

text

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

where

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

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

text

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

Chapter 4

Using the DRS

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

4.1 Running the DRS recipes directly

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

- To run from command line type:

CMD input

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

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

CMD input

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

- To run from ipython, open ipython and type:

Python/Ipython

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

4.2 Running the DRS recipes from a python script

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

Python/Ipython

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

4.3 Working example of the code for SPIRou

4.3.1 Overview

For this example all files are from:

CMD input

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

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

CMD input

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

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

1. cal_DARK_spirou.py (See Section 9.1)
2. cal_loc_RAW_spirou.py(×2) (See Section 9.2)
3. cal_SLIT_spirou.py (See Section 9.3)
4. cal_FF_RAW_spirou.py(×2) (See Section 9.4)
5. (add spirou_wave_ini3.fits to calibDB)
6. cal_extract_RAW_spirouAB.pyand cal_extract_RAW_spirouC.py(many times) (See Section 9.5)
7. cal_DRIFT_RAW_spirou.py (See Section 9.6)

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

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

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

   CMD input

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

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

   CMD input

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

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

CMD input

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

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

CMD input

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

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

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

text

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

to this:

text

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

CMD input

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

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

CMD input

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

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

Add the folloing line to 'master_calib_SPIROU.txt '

text

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

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

text

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

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

CMD input

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

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

CMD input

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

4.3.3 Run through python script

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

Python/Ipython

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

Chapter 5

Summary of changes (from version AT-4)

Chapter 6

Coding rules and standardisation practises

Chapter 7

Required input header keywords

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

7.1 Standard FITS Keywords

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

7.2 FITS keywords related to the detector

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

7.3 FITS keywords related to the target

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

7.4 FITS keywords related to the telescope

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

7.5 FITS keywords related to the instrument

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

Chapter 8

Variables

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

- **Variable title**

  Description of the variable

  VARIABLE_NAME = Default Value

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

  – Public: Everyone (including the user)

  – Private: Only the developer

8.1 Variable file locations

8.1.1 User modifiable variables

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

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

8.1.2 Private variables

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

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

  Python/Ipython

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

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

  KEYWORD = Default value / Comment

- **Constants and Pseudo-constants:** These are stored in SpirouDRS.spirouConfig.spirouConst, they range from simple objects (strings, integers, float, lists, python dictionaries etc) to more

complicated 'pseudo-constants' that are constructed themselves from other constants. These are kept private (i.e. no mentioned in the user manual) as they should not need be changed by the average user.

8.2 Global variables

- **DRS Name**

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

  DRS_NAME = SPIROU

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

- **DRS Version**

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

  DRS_VERSION = SPIROU

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

- **Plotting switch**

  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.txt
  Called in: All Recipes
  Level: Public

- **Debug mode**

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

  - 0 = No debug
  - 1 = Level 1 debug

    > **TODO:** Define level 1 debug

  - 2 = Level 2 debug

    > **TODO:** Define level 2 debug

  ic_debug = 0

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

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

- **Plot interval**

  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.py
  Defined in: constants_SPIROU.txt
  Called in:
  Level: Public

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

8.3 Directory variables

- **The data directory**

  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.txt
  Called in: SpirouDRS.spirouConfig.spirouConst
  Level: Public

- **The installation directory**

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

  DRS_ROOT = /drs/INTROOT/

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

- **The raw data directory**

  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.txt
  Called in: SpirouDRS.spirouConfig.spirouConst
  Level: Public

- **The reduced data directory**

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

  DRS_DATA_REDUC = /drs/data/reduced

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

- **The calibration database and calibration file directory**

  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.txt
  Called in: SpirouDRS.spirouConfig.spirouConst
  Level: Public

- **The log directory**

  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.txt
  Called in: SpirouDRS.spirouConfig.spirouConst
  Level: Public

- **The working directory**

  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.txt
  Called in: SpirouDRS.spirouConfig.spirouConst
  Level: Public

8.4 Image variables

- **Resizing blue window**

The blue window used in cal_DARK_spirou.py. 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.py
Defined in: constants_SPIROU.txt
Called in: cal_DARK_spirou.py.__main__()
Level: Public

- **Resizing red window**

The blue window used in cal_DARK_spirou.py. 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.py
Defined in: constants_SPIROU.txt
Called in: cal_DARK_spirou.py.__main__()
Level: Public

- **Resizing red window**

The blue window used in cal_DARK_spirou.py. 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.py, cal_SLIT_spirou.py, cal_FF_RAW_spirou.py, cal_extract_RAW_spirou.py, cal_DRIFT_RAW_spirou.py
Defined in: constants_SPIROU.txt
Called in: cal_loc_RAW_spirou.py__main__(), cal_SLIT_spirou.py__main__(), cal_FF_RAW_spirou.py__main__(), cal_extract_RAW_spirou.py__main__(), cal_DRIFT_RAW_spirou.py.__main__()
Level: Public

8.5 Fiber variables

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

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

- **Number of fibers**

  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.py
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.py.__main__()
  Level: Public

- **Order skip number**

  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.py
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.py.__main__()
  Level: Public

- **Maximum order numbers**

  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.py
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.py.__main__()
  Level: Public

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

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.py
Defined in: constants_SPIROU.txt
Called in: cal_loc_RAW_spirou.py.__main__()
Level: Public

Note: Should this be merged with 'ic_locnbmaxo_fpall'?

- **Fiber types for this fiber**

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.py
Defined in: constants_SPIROU.txt
Called in: cal_FF_RAW_spirou.py.__main__()
Level: Public

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

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

The pixels are extracted from the center of the order out to the edges - defined by 'top' and 'bottom' - width (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.py
Defined in: constants_SPIROU.txt
Called in: cal_extract_RAW_spirou.py.__main__(), SpirouDRS.spirouEXTOR.spirouEXTOR.extract_tilt_weight_order2()
Level: Public

Note: Formally this was called 'plage1' in cal_FF_RAW_spirou.py

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

The pixels are extracted from the center of the order out to the edges - defined by 'top' and 'botto'm - width (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.py, cal_extract_RAW_spirou.py
Defined in:	constants_SPIROU.txt
Called in:	cal_FF_RAW_spirou.py.__main__(), cal_extract_RAW_spirou.py.__main__(), SpirouDRS.spirouEXTOR.spirouEXTOR.extract_tilt_weight_order2()
Level:	Public

Note: Formally this was called 'plage2' in cal_FF_RAW_spirou.py

- **Half-zone extraction width for full extraction**

The pixels are extracted from the center of the order out to the edges - defined by the left and right - or top and bottom - width (illuminated part of the order). In cal_extract_RAW_spirou.pyboth 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.py
Defined in:	constants_SPIROU.txt
Called in:	SpirouDRS.spirouEXTOR.spirouEXTOR.extract_order(), SpirouDRS.spirouEXTOR.spirouEXTOR.extract_tilt_order(), SpirouDRS.spirouEXTOR.spirouEXTOR.extract_tilt_weight_order(), SpirouDRS.spirouEXTOR.spirouEXTOR.extract_weight_order()
Level:	Public

Note: Formally this was called 'plage' in cal_extract_RAW_spirou.py

- **Localization fiber for extraction**

  Defines the localization fiber to use for each fiber type. This is the file in calibDB that is used i.e. the keyword master_calib_SPIROU.txt used will be ÌOC_{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.py
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouLOCOR.spirouLOCOR.get_loc_coefficients()
  Level: Public

- **Order profile fiber for extraction**

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

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

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

- **Half-zone extract width cal_DRIFT_RAW_spirou.py**

  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.py
  Defined in: constants_SPIROU.txt
  Called in: cal_DRIFT_RAW_spirou.py.__main__()
  Level: Public

  Note: Formally this was called 'ic_extnbsig' in cal_DRIFT_RAW_spirou.py

8.6 Dark calibration variables

- **Lower percentile for dead pixel stats**

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

- **Upper percentile for dead pixel stats**

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

- **Dark stat histogram 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.py
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouImage.spirouImage.measure_dark()
  Level: Public

- **Lower bound for the Dark stat histogram**

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

  histo_range_low = -0.5

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

- **Upper bound for the Dark stat histogram**

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

- **Bad pixel cut limit**

  Defines the bad pixel cut limit in ADU/s.

  $$badpixels = (image > \text{dark\_cut\_limit}) \text{ OR (non-finite)} \tag{8.1}$$

  dark_cutlimit = 100.0

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

8.7 Localization calibration variables

- **Order profile smoothed 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.py
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.py.__main__()
  Level: Public

- **Image row 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.py
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.py.__main__()
  Level: Public

- **Central column of the image**

  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.py
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.py.__main__(), SpirouDRS.spirouLOCOR.spirouLOCOR.find_order_centers(), SpirouDRS.spirouLOCOR.spirouLOCOR.initial_order_fit()
  Level: Public

- **Localization window row size**

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

  Note: Formally this was called 'ic_ccdcolc' in cal_loc_RAW_spirou.py

- **Localization window column step**

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

- **Image gap index**

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

  $$\text{(top of the row} < \text{ic\_image\_gap) OR (bottom of the row} > \text{ic\_image\_gap)} \tag{8.2}$$

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

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

- **Minimum order row size**

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

- **Min/Max smoothing box size**

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

- **Minimum signal 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.py
  Defined in: constants_SPIROU.txt
  Called in: SpirouDRS.spirouBACK.spirouBACK.measure_background_and_get_central_pixels()
  Level: Public

- **Normalized background amplitude threshold**

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

- **Saturation threshold on the order profile plot**

  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.py
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.py.__main__()
  Level: Public

- **Degree of the fitting polynomial for localization position**

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

- **Degree of the fitting polynomial for localization width**

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

- **Degree of the fitting polynomial for localization position**

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

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

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

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

$$max(RMS) > \text{ic\_max\_rms\_center} \tag{8.3}$$

ic_max_rms_center = 0.2

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

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

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

  $$max(|\text{residuals}|) > \text{ic\_max\_ptp\_center} \tag{8.4}$$

  ic_max_ptp_center = 0.2

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

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

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

  fit is recalculated if:

  $$max(|\text{residuals}|)/\text{RMS} > \text{ic\_ptporms\_center} \tag{8.5}$$

  ic_ptporms_center = 8.0

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

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

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

  $$max(RMS) > \text{ic\_max\_rms\_width} \tag{8.6}$$

  ic_max_rms_fwhm = 1.0

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

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

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

  $$max(|\text{residuals/data}|) \times 100 > \text{ic\_max\_ptp\_fracfwhm} \tag{8.7}$$

  ic_max_ptp_fracfwhm = 1.0

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

- **Delta width 3 convolve shape model**

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

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

- **Localization archiving option**

  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.py
  Defined in: constants_SPIROU.txt
  Called in: cal_loc_RAW_spirou.py.__main__()
  Level: Public

8.8 Slit calibration variables

- Items here

8.9 Flat fielding calibration variables

- Items here

8.10 Extraction calibration variables

- Items here

8.11 Drift calibration variables

- Items here

8.12 Quality control variables

- Items here

8.13 Formatting variables

- Items here

8.14 Logging and printing variables

- **Print message 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.txt
  Called in:
  Level: Public

- **Log message 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.txt
  Called in:
  Level: Public

Chapter 9

The Recipes

Chapter 10

The DRS Module