SPIRou User Manual

Version 0.0.1

N. Cook

2017-10-19

Abstract

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

Contents

Introduction

Some text here

Notes

- This installation assumes you are running bash on a Linux machine.
- There is no need for root access needed for any of these steps.
- The python modules required for this DRS to run (more up-to-date versions are NOT supported) will make current versions of python not work. It is currently recommended that an isolated version of python be used with this version of the DRS (see Section 1.5).
- Variables inside { } are defined in Section 1.2 and used throughout to mean "replace with valued defined in Section 1.2".
- The following denotes a line of text that is to be edited (in a file):

```
VARIABLE_NAME="Variable Value"
```

- The following denotes a console command to run:

```
echo "HelloWorld!"
```

- The following denotes a python command to run:

```
>>> print("HelloWorld!")
```

Chapter 1

Pre-Installation

This file is just for use with the current installation files (version 43), the author does not recommend that this is the final set-up procedure, nor that any of the 'modifications' to the original source code applied here should be used in any final version of the DRS (better solutions should be found).

1.1 Prerequisites

Before one can use the DRS pipeline one must follow the followings steps before installing the pipeline.

1. Add variables to the system environment (Section 1.2)
2. Setup and check the folder structure (Section 1.3)
3. Make sure Fortran and C back-ends are installed (Section 1.4)
4. Install an isolated, clean version of python (Section 1.5)
5. Install required python modules - exact version not older not newer (Section 1.6)

There is no need for root access for any of the following steps.

1.2 Adding variables to the system environment

This can be done one of two ways. The first root (Section 1.2.1) is via a setup file and will allow you to switch on and switch of the environmental variables (to avoid clashes with other programs and to keep ones environment clean). Note with this method you will have to source the setup script before installing or running this code (each time the environment changes) or add the activation to your bashrc. This route is recommended. Otherwise you can follow the second root (Section 1.2.2) and hard-code variables to your '.bashrc' file.

1.2.1 Using a setup script

Open up 'env_setup.sh' and edit the following lines. You will find a copy of 'env_setup.sh' in Appendix A.

1. Set the instrument name {INSTRUMENT_NAME}
2. Directory in which all SPIROU files go {DATA_ROOT}
3. Define installation folder name {INSTALL_ROOT}
4. Define data folder name {DATA_ROOT}
5. Define raw path {DATA_ROOT_RAW}
6. Define reduced path {DATA_ROOT_REDUCED}
7. Define calibDB path {DATA_ROOT_CALIB}
8. Define msg path {DATA_ROOT_MSG}
9. Define tmp path {DATA_ROOT_TMP}
10. Define python version {PYTHON_VERSION}
11. Define python directory (i.e. result of command "which python") {PYTHON_DIR}
12. Define GSL path (default paths if installed may be /usr/local/include/gsl or /opt/gsl) {GSL_DIR}

i.e. these lines in 'env_setup.sh'.

```
INSTRUMENT_NAME="SPIROU"
DIR="/data/spirou/drs"
INSTALL_ROOT="$DIR/INTROOT"
DATA_ROOT="$DIR/data"
DATA_RAW_ROOT="$DATA_ROOT/raw/"
DATA_ROOT_REDUCED="$DATA_ROOT/reduced/"
DATA_ROOT_CALIB="$DATA_ROOT/calibDB"
DATA_ROOT_MSG="$DATA_ROOT/msg/"
DATA_ROOT_TMP="$DATA_ROOT/tmp/"
PYTHON_VERSION="2.7"
PYTHON_DIR="$DIR/python/miniconda2/"
GSL_DIR="$DIR/c-libraries/gsl"
```

where any pre-defined variable can be used with a preceding $ sign (to avoid repetition - coloured above in red).

1.2.2 Adding variables to .bashrc file

Open '~.bashrc' in your favourite text editor.

Add the following to it:

```
export INSTRUMENT={INSTRUMENT_NAME}
export DRS_DATA_RAW={DATA_ROOT_RAW}
export DRS_DATA_REDUC={DATA_ROOT_REDUCED}
export DRS_CALIB_DB={DATA_ROOT_CALIB}
export DRS_DATA_MSG={DATA_ROOT_MSG}
export DRS_DATA_WORKING={DATA_ROOT_TMP}
export TDATA={DATA_ROOT}

export INTROOT={INSTALL_ROOT}
export PATH={INSTALL_ROOT}/bin:{PYTHON_DIR}/bin/:$PATH
export PYTHONPATH=.:{PYTHON_DIR}/lib/python{PYTHON_VERSION}/site-packages/:{INSTALL_ROOT}/bin
export DRS_LOG=1
export PYTHON_INCLUDE_DIR="{PYTHON_DIR}/lib/python{PYTHON_VERSION}/site-packages/numpy/core/
   include"
export GSL_INCLUDE_DIR={GSL_DIR}/include
export GSL_LIBRARY_DIR={GSL_DIR}/lib
```

where:

- {INSTRUMENT_NAME} = Set the instrument name
- {DATA_ROOT}= Directory in which all SPIROU files go
- {INSTALL_ROOT} = Define installation folder name
- {DATA_ROOT} = Define data folder name
- {DATA_ROOT_RAW} = Define raw path
- {DATA_ROOT_REDUCED} Define reduced path
- {DATA_ROOT_CALIB} = Define calibDB path
- {DATA_ROOT_MSG} = Define msg path
- {DATA_ROOT_TMP} = Define tmp path
- {PYTHON_VERSION} = Define python version
- {PYTHON_DIR} = Define python directory (i.e. result of command "which python")
- {GSL_DIR} = Define GSL path (default paths if installed may be /usr/local/include/gsl or /opt/gsl)

1.3 Setup up and check file structure

Make sure the following folders are created:

```
mkdir {DIR}
mkdir {DIR}/{INSTALL_ROOT}/bin
mkdir {DATA_ROOT}
mkdir {DATA_RAW_ROOT}
mkdir {DATA_ROOT_REDUCED}
mkdir {DATA_ROOT_CALIB}
mkdir {DATA_ROOT_MSG}
mkdir {DATA_ROOT_TMP}
```

i.e. for the above 'env_setup.sh' this would be

```
mkdir "/data/spirou/drs"
mkdir "/data/spirou/drs/INTROOT/bin"
mkdir "/data/spirou/drs/data"
mkdir "/data/spirou/drs/data/raw"
mkdir "/data/spirou/drs/data/reduced"
mkdir "/data/spirou/drs/data/calibDB"
mkdir "/data/spirou/drs/data/msg"
mkdir "/data/spirou/drs/data/tmp"
```

1.4 Fortran and C back-end installation

1.4.1 Checking Fortran/C back-ends

!!!! TODO !!!! design checks for Fortran and C, or do we just say that installation requires Fortran and C to be install
(Installing Fortran and C will require root access). !!!!

Please check whether Fortran and C are installed. In addition to C you will need the C package 'GSL', check that it is installed (default paths may be: /usr/local/include/gsl or /opt/gsl). If it is not installed please follow the instructions in Section 1.4.2.

1.4.2 Installing GSL back-ends without root access

To install GSL without root us the following steps:

1. download from http://ftpmirror.gnu.org/gsl/
   ```
   wget http://ftpmirror.gnu.org/gsl/gsl-1.16.tar.gz
   ```
2. create the {GSL_DIR} (from above)
   ```
   mkdir {GSL_DIR}
   ```
3. untar the GSL files
   ```
   tar -xvf gs1-1.16.tar.gz
   ```
4. change to untarred directory
   ```
   cd gs1-1.16/
   ```
5. configure the installation dir for GSL
   ```
   ./configure prefix={GSL_DIR}
   ```
6. build the GSL installation
   ```
   make
   ```
7. install the GSL installation
   ```
   make install
   ```

1.5 Install isolated version of Python

As the current DRS requires specific versions of modules to run we recommend a isolated version of python (as to not interfere with your system or running other python codes). Python installation must meet the following specifications:

- numpy 1.8.2 (versions later than 1.8.2 are unsupported)
- scipy 0.14
- matplotlib 1.3.1
- pyfits 3.2.4

Hence installing Miniconda (a minimal version of the anaconda python distribution) is the easiest way to achieve this in an isolated environment (as not to destroy any current/system version of python). See section 1.5.1 for Miniconda install instructions.

1.5.1 Installing Miniconda

To install Miniconda follow the steps below:

1. download miniconda from here: https://conda.io/miniconda.html

   ```
   wget https://repo.continuum.io/miniconda/Miniconda2-latest-Linux-x86_64.sh
   ```

2. run bash script

   ```
   bash Miniconda2-latest-Linux-x86_64.sh
   ```

   Note: choose Miniconda installation directory to match {PYTHON_DIR} above
   Note: if you wish to activate this environment/use other python installations do not add Miniconda2 install location to PATH in ~/.bashrc

3. add {PYTHON_DIR} to FRONT of PATH environment (temporarily).

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

4. check that we are using the correct version of python/conda/pip

   ```
   which python
   ```

   Should read:

   ```
   {PYTHON_DIR}/bin/python
   ```

   or

   ```
   {PYTHON_DIR}/bin/python{PYTHON_VERSION}
   ```

   similarly for

   ```
   which conda
   which pip
   ```

which should read:

```
{PYTHON_DIR}/bin/conda
```

and

```
{PYTHON_DIR}/bin/pip respectively
```

1.6 Install required python modules

1.6.1 Installation

1. install numpy with miniconda

```
conda install numpy==1.8.2
```

```
Fetching package metadata .........
Solving package specifications: .

Package plan for installation in environment /scratch/bin/miniconda2/test:

The following NEW packages will be INSTALLED:

    libgfortran: 1.0-0
    numpy:       1.8.2-py27_1

The following packages will be UPDATED:

    conda:       4.3.21-py27_0 --> 4.3.27-py27hff99c7a_0

          Proceed ([y]/n)? y
```

2. install scipy with miniconda

```
conda install scipy==0.14
```

```
Fetching package metadata ...........
Solving package specifications: .

Package plan for installation in environment /scratch/bin/miniconda2/test:

The following NEW packages will be INSTALLED:

    scipy:     0.14.0-np18py27_0

The following packages will be UPDATED:

    conda-env: 2.6.0-0           --> 2.6.0-h36134e3_1


          Proceed ([y]/n)? y
```

3. install matplotlib with miniconda

```
conda install matplotlib==1.3.1
```

```
Fetching package metadata ...........
Solving package specifications: .

Package plan for installation in environment /scratch/bin/miniconda2/test:
```

```
The following NEW packages will be INSTALLED:

    cairo:      1.12.18-0
    dateutil:   2.4.1-py27_0
    freetype:   2.4.10-0
    libpng:     1.5.13-1
    matplotlib: 1.3.1-np18py27_1
    pixman:     0.26.2-0
    py2cairo:   1.10.0-py27_2
    pyqt:       4.10.4-py27_0
    pytz:       2017.2-py27hcac29fa_1
    qt:         4.8.5-0
    sip:        4.15.5-py27_0

The following packages will be DOWNGRADED:

    pyparsing:  2.1.4-py27_0          --> 2.0.1-py27_0


        Proceed ([ y ]/ n ) ? y
```

4. download pyfits 3.2.4 from http://www.stsci.edu/institute/software_hardware/pyfits/Download

```
wget https://pypi.python.org/packages/source/p/pyfits/pyfits-3.2.4.tar.gz
```

5. install pyfits with pip

```
pip install pyfits-3.2.4.tar.gz
```

1.6.2 Checking installed module versions

Before we continue we should check python and module installation.

1. run python

```
python
```

Inside python run following commands

```
>>> import numpy
>>> import matplotlib
>>> import scipy
>>> import pyfits
```

Test the version with:

```
>>> numpy.__version__
1.8.2
>>> matplotlib.__version__
1.3.1
>>> scipy.__version__
0.14.0
>>> pyfits.__version__
3.2.4
```

Chapter 2

Installation

This is just for use with the current installation files (version 43), the author does not recommend that this is the final set-up procedure, nor that any of the 'modifications' to the original source code applied here should be used in any final version of the DRS (better solutions should be found).

2.1 Activate environment

If you followed the steps in 1.2.1 please follow this section, if however you followed the steps in 1.2.2 continue to Section 2.2.

Before running the installation (or before running the code) one must run the following:

```
source env_setup.sh
```

Output should look like this:

```
  ========================================
    Environmental setup for SPIRou Pipeline
  ========================================
    Setting up environment
    Set up for {INSTRUMENT}
      - Python located at: {PYTHON_DIR}
      - GLS located at: {GSL_DIR}
      - data located at:
            {DATA_ROOT}
            {DATA_RAW_ROOT}
            {DATA_ROOT_REDUCED}
            {DATA_ROOT_CALIB}
            {DATA_ROOT_MSG}
            {DATA_ROOT_TMP}

    Done
```

Note to deactivate type

```
source env_setup.sh --clean
```

2.2 Downloading the install scripts

Minor modifications need to be made to the code to allow a isolated version of GSL to be used. If and only if your GSL is installed to '/opt/gsl/' will the code install correctly with out this modification.

The other steps are as in the original installation procedure.

1. change directory to {DIR}

```
{DIR}
```

2. download source code for spirou, location should be {DIR}/spirou

```
svn co https://svn.lam.fr/repos/spirou/trunk spirou
```

1. change directory to source files

```
cd {DIR}/spirou/src
```

2. find and change {DIR}/spirou/src/C/setup.py

   a) Below the line {PYTHON_INCLUDE_DIR} = os.getenv('{PYTHON_INCLUDE_DIR}') add the following:

```
gsl_include_dir = os.getenv('GSL_INCLUDE_DIR')
gsl_library_dir = os.getenv('GSL_LIBRARY_DIR')
```

   b) Find and replace all instances of

```
include_dirs = [python_include_dir,'/opt/gsl/include'],
```

   with

```
include_dirs = [python_include_dir, gsl_include_dir],
```

   c) Find and replace all instances of

```
library_dirs = ['/opt/gsl/lib'],
```

   with

```
library_dirs = [gsl_library_dir],
```

2.3 Running installation scripts

run installation scripts

```
./hardrsInstall SPIROU 2.7
./scriptInstall SPIROU 2.7
```

2.4 Updating local version

The code currently uses 'Subversion' (SVN) to monitor the different versions of the code that is under development. To update a local version of the DRS:

1. Go inside the {DIR}/spirou folder:
   ```
   cd \{DIR\}/spirou
   ```
2. Type:
   ```
   svn update
   ```
   You will then receive information on the DRS version.
3. Then you need to re-install the DRS. This will rewrite all codes in the {INSTALL_ROOT} folder.
   Reinstall the DRS using:
   ```
   cd src
   ./hardrsInstall SPIROU 2.7
   ./scriptInstall SPIROU 2.7
   ```

You are now ready to work with the latest version.

2.5 Adding new files to the shared DRS

Note: this is only to be done if your new functions are running correctly.

1. Go inside the {DIR}/spirou folder:
   ```
   cd \{DIR\}/spirou
   ```
2. run the update on the svn
   ```
   svn update
   ```
3. copy your modifications into the source code:
   ```
   cp {modified file} src/{location of new file}
   ```
4. Add the file to the list of pending SVN files:
   ```
   svn add src/{location of new file}
   ```
5. once all new files are added, commit to releasing the modifications to the SVN:
   ```
   svn commit -m 'modification of {file name}'
   ```

2.5.1 Useful links about subversion

:

Presentation: http://multithread.org/files/presentations/subversion/subversion_tutorial.pdf Book: http://svnbook.red-bean.com/ Introduction to subversion: https://dev.nozav.org/intro_svn.html

Chapter 3

Using the DRS

3.1 Running the code

To run the DRS python (assuming env_setup is activated, see Section 2.1) type:

```
DRS_{INSTRUMENT} -m
```

i.e.

```
DRS_spirou -m {PROGRAM NAME} {FOLDER} {FILES}
DRS_spirou -m cal_DARK_spirou {YYMMDD} {Filenames*}
```

instead of python
Note: location should of script should be {DIR}/{INSTALL_ROOT}/bin/DRS{INSTRUMENT}
i.e. for 'cal_DARK_spirou' in the {DATA_RAW_ROOT}/YYMMDD directory one would result in something like the following:

```
DRS_spirou -m cal_DARK_spirou YYMMDD Filenames*
```

```
19:44:52.5 -   || ****************************************
19:44:52.5 -   || * SPIROU (@) Geneva Observatory ()
19:44:52.5 -   || ****************************************
19:44:52.5 -   ||(dir_data_raw)      DRS_DATA_RAW=/data/spirou/drs/data/raw/
19:44:52.5 -   ||(dir_data_reduc)    DRS_DATA_REDUC=/data/spirou/drs/data/reduced/
19:44:52.5 -   ||(dir_drs_config)    DRS_CONFIG=/data/spirou/drs/INTROOT/DRS_SPIROU/config/
19:44:52.5 -   ||(dir_calib_db)      DRS_CALIB_DB=/data/spirou/drs/data/calibDB
19:44:52.5 -   ||(dir_data_msg)      DRS_DATA_MSG=/data/spirou/drs/data/msg/
19:44:52.5 -   ||(print_log)         DRS_LOG=1         %(0: minimum stdin-out logs)
19:44:52.5 -   ||(plot_graph)        DRS_PLOT=NONE            %(def/undef/trigger)
19:44:52.5 -   ||(used_date)         DRS_USED_DATE=undefined
19:44:52.5 -   ||(working_dir)       DRS_DATA_WORKING=/data/spirou/drs/data/tmp/
19:44:52.5 -   ||                    DRS_INTERACTIVE is not set, running on-line mode
19:44:52.5 -   |-c:+[...]|Now running : -c on file(s):  dark_dark...
19:44:52.5 -   |-c:+[...]|On directory /data/spirou/drs/data/raw/20170811
19:44:52.5 -   |-c:+[...]|ICDP loaded from: /data/spirou/drs/INTROOT/DRS_SPIROU/config/
    hadmrICDP_SPIROU.py
19:44:52.5 - * |-c:+[...]|Now processing Image TYPE DARK with -c recipe
19:44:52.5 -   |-c:+[...]|Reading Image /data/spirou/drs/data/raw/20170811/dark_dark02d.fits
19:44:52.5 -   |-c:+[...]|Image 2048x2048 loaded
```

3.2 Working example of the code for SPIRou

All files are from:

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

Will also need current WAVE file from here:

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

Setup as in previous sections.

1. Change to the {DIR} directory

   ```
   cd DIR
   ```

2. Activate the environment (in bash only)

   ```
   source env_setup.sh
   ```

3. Check that environment is active

   ```
   echo $DRS_ACTIVE
   ```

   result should be = 1

4. run the dark extraction:

   ```
   DRS_spirou -m cal_DARK_spirou.py 20170710 dark_dark02d406.fits
   ```

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

   ```
   DRS_spirou -m cal_DARK_spirou.py 20170710 dark_dark02d406.fits
   ```

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

   ```
   DRS_spirou -m cal_loc_RAW_spirou.py 20170710 dark_flat02f10.fits dark_flat03f10.fits
       dark_flat04f10.fits dark_flat05f10.fits dark_flat06f10.fits
   ```

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

   ```
   DRS_spirou -m cal_loc_RAW_spirou.py 20170710 flat_dark02f10.fits flat_dark03f10.fits
       flat_dark04f10.fits flat_dark05f10.fits flat_dark06f10.fits
   ```

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

   ```
   DRS_spirou -m cal_SLIT_spirou.py 20170710 fp_fp02a203.fits fp_fp03a203.fits fp_fp04a203.
       fits
   ```

9. 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 the unix date on rwos 'ORDER_PROFIL_AB' (i.e. change it from 1499707515.0 to 1499705515.0).

   i.e. the 'master_calib_SPIROU.txt' file should look go from

```
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:

```
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
```

```
DRS_spirou -m cal_FF_RAW_spirou.py 20170710 dark_flat02f10.fits dark_flat03f10.fits
    dark_flat04f10.fits dark_flat05f10.fits dark_flat06f10.fits
```

10. run the flat field creation on the 'flat_dark' files:

```
DRS_spirou -m cal_FF_RAW_spirou.py 20170710 flat_dark02f10.fits flat_dark03f10.fits
    flat_dark04f10.fits flat_dark05f10.fits flat_dark06f10.fits
```

11. Currently we do not create a new wavelength calibration file for this run. Therefore we need to get one from here:

```
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'

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

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

```
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
```

12. 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:

```
DRS_spirou -m cal_extract_RAW_spirouAB.py 20170710 hcone_dark02c61.fits  hcone_dark03c61.
    fits  hcone_dark04c61.fits  hcone_dark05c61.fits  hcone_dark06c61.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouC.py 20170710 dark_hcone02c61.fits  dark_hcone03c61.
    fits  dark_hcone04c61.fits  dark_hcone05c61.fits  dark_hcone06c61.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouAB.py 20170710 hcone_hcone02c61.fits  hcone_hcone03c61.
    fits  hcone_hcone04c61.fits  hcone_hcone05c61.fits  hcone_hcone06c61.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouC.py 20170710 hcone_hcone02c61.fits  hcone_hcone03c61.
    fits  hcone_hcone04c61.fits  hcone_hcone05c61.fits  hcone_hcone06c61.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouAB.py 20170710 dark_dark_AHC102d61.fits
    dark_dark_AHC103d61.fits  dark_dark_AHC104d61.fits  dark_dark_AHC105d61.fits
    dark_dark_AHC106d61.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouC.py 20170710 dark_dark_AHC102d61.fits
    dark_dark_AHC103d61.fits  dark_dark_AHC104d61.fits  dark_dark_AHC105d61.fits
    dark_dark_AHC106d61.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouAB.py 20170710 hctwo_dark02c61.fits  hctwo_dark03c61.
    fits  hctwo_dark04c61.fits  hctwo_dark05c61.fits  hctwo_dark06c61.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouC.py 20170710 hctwo_dark02c61.fits  hctwo_dark03c61.
    fits  hctwo_dark04c61.fits  hctwo_dark05c61.fits  hctwo_dark06c61.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouAB.py 20170710 dark_hctwo02c61.fits  dark_hctwo03c61.
    fits  dark_hctwo04c61.fits  dark_hctwo05c61.fits  dark_hctwo06c61.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouC.py 20170710 dark_hctwo02c61.fits  dark_hctwo03c61.
    fits  dark_hctwo04c61.fits  dark_hctwo05c61.fits  dark_hctwo06c61.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouAB.py 20170710 hctwo-hctwo02c61.fits  hctwo-hctwo03c61.
    fits  hctwo-hctwo04c61.fits  hctwo-hctwo05c61.fits  hctwo-hctwo06c61.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouC.py 20170710 hctwo-hctwo02c61.fits  hctwo-hctwo03c61.
    fits  hctwo-hctwo04c61.fits  hctwo-hctwo05c61.fits  hctwo-hctwo06c61.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouAB.py 20170710 dark_dark_AHC202d61.fits
    dark_dark_AHC203d61.fits  dark_dark_AHC204d61.fits  dark_dark_AHC205d61.fits
    dark_dark_AHC206d61.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouC.py 20170710 dark_dark_AHC202d61.fits
    dark_dark_AHC203d61.fits  dark_dark_AHC204d61.fits  dark_dark_AHC205d61.fits
    dark_dark_AHC206d61.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouAB.py 20170710 fp_fp02a203.fits fp_fp03a203.fits
    fp_fp04a203.fits
```

```
DRS_spirou -m cal_extract_RAW_spirouC.py 20170710 fp_fp02a203.fits fp_fp03a203.fits
    fp_fp04a203.fits
```

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

```
DRS_spirou -m cal_DRIFT_RAW_spirou.py 20170710 fp_fp02a203.fits fp_fp03a203.fits
    fp_fp04a203.fits
```

Chapter 4

Data Architecture

Below we describe the data architecture, first by providing the installed file structure in a tree diagram (4.1), and then describing individual directories.

4.1 Installed file structure

The file structure should look as follows:

```
{DIR}
├── {INSTALL_ROOT}
│   ├── bin
│   │   └── ..................................................Recipe symbolic-link
│   └── DRS_{INSTRUMENT_NAME}
│       └── ..................................................Worker functions
├── spirou
│   ├── read_harps.py
│   └── src
│       └── ..............................................source installation files
└── {DATA_ROOT}*
    ├── raw
    │   └── YYYYMMDD .....................................Observation directory
    │       └── ..........................................Raw observation files
    ├── reduced
    ├── calibDB
    ├── msg
    └── tmp
```

* Note that 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 1.2.

i.e. for the given 'env_setup.sh' (Appendix A) this would be:

```
/data/spirou/drs
├── INTROOT/
│   ├── bin
│   │   └── ..................................... Recipe symbolic-link
│   └── DRS_SPIROU
│       └── ..................................... Worker functions
├── data
│   ├── raw
│   │   └── YYYYMMDD ..................... night_repository
│   │       └── ..................... Raw observation files
│   ├── reduced
│   ├── calibDB
│   ├── msg
│   └── tmp
└── spirou
    ├── read_harps.py
    └── src
        └── ..................... source installation files
```

4.2 The {INSTALL_ROOT} directory

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

```
/data/spirou/drs
└── {INSTALL_ROOT}
    ├── bin
    │   └── ..................... Recipes
    └── DRS_SPIROU
```

4.2.1 The bin directory

The bin directory is located in the {INSTALL_ROOT} directory. This contains symbolic links to the DRS recipes.

4.2.2 The DRS_SPIROU directory

The DRS_SPIROU directory contains all the worker functions and configuration files for the DRS. The file structure is as follows:

```
DRS_SPIROU
├── C
│   └── ..........................................The C programs and setup.py
├── config
│   └── ..................................................The config files
├── docs
├── fortran
│   └── ..............................The fortran programs and install.csh
├── man
├── python
│   ├── C-modules
│   ├── f2pymodules
│   ├── Modules
│   └── Recipes
└── util
```

The C and Fortran functions are found in their own directories. The configuration files are explained in Section 4.5.

The python directory contains the translation of the C and fortran functions into python (C-modules and f2pymodules directories).

The functions 'startup.py' and 'startup_recipes.py' are called automatically by the recipes.

'python.csh' is the custom python environment for the DRS.

The python modules are explained in Section 5 and the recipes are explained in Section 6.

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 {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 1.2 these sub-directories can be defined elsewhere.

4.3.1 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
```

4.4 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 qulaity tests and a test file 'master_calib.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 uniqie calibration file and lines are formatted in the following manner:

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

where

- {key} is a code assigned for each type of calibration file. Currently accepted keys are:
  - DARK - Created from cal_DARK_spirou
  - ORDER_PROFIL_{fiber} - Created in cal_loc_RAW_spirou
  - LOC_C - Created in cal_loc_RAW_spirou
  - TILT - Created in cal_SLIT_spirou
  - FLAT_{fiber} - Created in cal_FF_RAW_spirou
  - WAVE - Currently manually added
- {night_repository} is the raw data observation directory (in {DATA_ROOT_RAW}) normally in the form YYYYMMDD.

- {filename} is the filename of the calibration file (located in the calibDB).
- {human readable date} is the date in DD/MM/YY/HH:MM:SS format taken from the header of the file that created the calibration file (using the header keyword 'ACQTIME1').
- {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})

```
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
```

4.5 The configuration files

Some text here.

Chapter 5

The Modules

Some text here

5.1 The Module directory

This contains all the python worker files for the DRS. The file structure is as below.

```
{DIR}
└── {INSTALL_ROOT}
    └── DRS_SPIROU
        └── python
            └── Modules
                ├── hadgtVISU
                │   └── ..................................hadmrVISU module files
                ├── hadmrBACK
                │   └── ..................................hadmrBACK module files
                ├── hadmrBIAS
                │   └── ..................................hadmrBIAS module files
                ├── hadmrCDB
                │   └── ..................................hadmrCDB module files
                ├── hadmrDARK
                │   └── ..................................hadmrDARK module files
                ├── hadmrEXTOR
                │   └── .................................hadmrEXTOR module files
                ├── hadmrFITS
                │   └── ..................................hadmrFITS module files
                ├── hadmrFLAT
                │   └── ..................................hadmrFLAT module files
                ├── hadmrLED
                │   └── ..................................hadmrLED module files
                ├── hadmrLOCOR
                │   └── .................................hadmrLOCOR module files
```

5.2 spirouBACK

spirouBACK.py contains functions to calculate the detector background

5.3 spirouCDB

spirouCDB.py: contains functions for the calibrations database (infos for master_calib.txt, and to coy files in the caliDB directory)

5.4 spirouEXTOR

spirouEXTOR.py: contains function for the ordres extraction

5.5 spirouFITS

spirouFITS.py: contains functions to read FITS file and copy keywords

5.6 spirouLOCOR

spirouLOCOR.py: contains functions for order localization

5.7 spirouRV

spirouRV.py: contains functions to compute radial velocity

5.8 spirouVISU

spirouVISU.py: contains functions for visualization

Chapter 6

The Recipes

6.1 The Recipe directory

These are symbolically linked under {INSTALL_ROOT}/bin. A list of all current recipes is below:

```
{DIR}
└── {INSTALL_ROOT}
    └── DRS_SPIROU
        └── python
            └── Recipes
                ├── cal_BIAS_spirou.py
                ├── cal_CONTAM_spirou.py
                ├── cal_DARK_spirou.py
                ├── cal_DRIFT_E2DS_spirou.py
                ├── cal_DRIFT_RAW_spirou.py
                ├── cal_extract_RAW_spirouAB.py
                ├── cal_extract_RAW_spirouALL.py
                ├── cal_extract_RAW_spirouC.py
                ├── cal_FFPOL_spirou.py
                ├── cal_FF_RAW_spirou.py
                ├── cal_FF_spirou.py
                ├── cal_loc_ONE_spirou.py
                ├── cal_loc_RAW_spirou.py
                ├── cal_SLIT_spirou.py
                ├── cal_TH_spirou.py
                ├── cal_WAVE_spirou.py
                ├── db_get_files_spirou.py
                ├── db_reduce_star_spirou.py
                ├── db_update_spirou.py
                ├── Install.csh
                └── mai_cal_drift_spirou.py
```

```
├── mai_compute_drift_spirou.py
├── mai_config_wavecal_spirou.py
├── mai_make_flux_template_spirou.py
├── mai_plot_drift_spirou.py
├── obj_ONE_spirou.py
├── obj_TH_spirou.py
├── obj_TWO_spirou.py
├── obj_WAVE_spirou.py
├── off_make_bis_spirou.py
├── off_make_ccf_spirou.py
├── off_make_execCAL_spirou.py
├── off_make_execOBJ_spirou.py
├── off_make_exec_spirou.py
├── off_make_S_spirou.py
├── off_visu_bis_spirou.py
├── off_visu_ccf_spirou.py
├── off_visu_dark_spirou.py
├── off_visu_e2ds_spirou.py
├── off_visu_rvo_spirou.py
├── off_visu_s1d_spirou.py
├── off_visu_SN_spirou.py
├── ske_recipe_spirou.py
├── test_cal_loc_ONE_spirou.py
└── visu_RAW_spirou.py
```

6.2 Currently used

6.2.1 cal_DARK_spirou

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 automaticaly by the pipeline.

```
cal_DARK_spirou.py  night_repository  filename
```

An example run where everything worked is below:

```
DRS_spirou -m cal_DARK_spirou.py 20170710 dark_dark02d406.fits


20:44:08.3 -   ||DRS  SPIROU   v   (interactive mode)
20:44:08.3 -   || ****************************************
20:44:08.3 -   || * SPIROU @(#) Geneva Observatory ()
20:44:08.3 -   || ****************************************
20:44:08.3 -   ||(dir_data_raw)      DRS_DATA_RAW=/scratch/Projects/SPIRou_Pipeline/data/raw/
20:44:08.3 -   ||(dir_data_reduc)    DRS_DATA_REDUC=/scratch/Projects/SPIRou_Pipeline/data/reduced
    /
20:44:08.3 -   ||(dir_drs_config)    DRS_CONFIG=/scratch/Projects/SPIRou_Pipeline/INTROOT/
    DRS_SPIROU/config/
```

```
20:44:08.3 -   ||(dir_calib_db)      DRS_CALIB_DB=/scratch/Projects/SPIRou_Pipeline/data/calibDB
20:44:08.3 -   ||(dir_data_msg)      DRS_DATA_MSG=/scratch/Projects/SPIRou_Pipeline/data/msg/
20:44:08.3 -   ||(print_log)         DRS_LOG=1         %(0: minimum stdin-out logs)
20:44:08.3 -   ||(plot_graph)        DRS_PLOT=NONE           %(def/undef/trigger)
20:44:08.3 -   ||(used_date)         DRS_USED_DATE=undefined
20:44:08.3 -   ||(working_dir)       DRS_DATA_WORKING=/scratch/Projects/SPIRou_Pipeline/data/tmp/
20:44:08.3 -   ||                    DRS_INTERACTIVE is  set
20:44:08.3 -   |-c:|Now running : -c on file(s):  dark_dark02d406.fits
20:44:08.3 -   |-c:|On directory /scratch/Projects/SPIRou_Pipeline/data/raw/20170710
20:44:08.3 -   |-c:|ICDP loaded from: /scratch/Projects/SPIRou_Pipeline/INTROOT/DRS_SPIROU/config/
    hadmrICDP_SPIROU.py
20:44:08.3 - * |-c:|Now processing Image TYPE DARK with -c recipe
20:44:08.3 -   |-c:|Reading Image /scratch/Projects/SPIRou_Pipeline/data/raw/20170710/
    dark_dark02d406.fits
20:44:08.4 -   |-c:|Image 2048x2048 loaded
20:44:08.6 - * |-c:|Dark Time = 597.489 [s]
20:44:08.8 -   |-c:|Doing Dark measurement
20:44:10.1 - * |-c:|Whole det   : Frac dead pixels= 14.7 % - Median= 0.35 ADU/s - Percent[5:95]=
    0.08-99.57 ADU/s
20:44:10.2 - * |-c:|In Blue part: Frac dead pixels= 1.0 % - Median= 0.15 ADU/s - Percent[5:95]=
    0.09-0.53 ADU/s
20:44:10.3 - * |-c:|In Red part : Frac dead pixels= 20.5 % - Median= 2.11 ADU/s - Percent[5:95]=
    0.18-232.09 ADU/s
20:44:10.4 - * |-c:|Total Frac dead pixels (N.A.N) + DARK > 100.0 ADU/s = 18.9 %
20:44:11.1 - * |-c:|QUALITY CONTROL SUCCESSFUL - Well Done -
20:44:11.1 -   |-c:|Saving Dark frame in dark_dark02d406.fits
20:44:12.2 -   |-c:|Saving Bad Pixel Map in dark_dark02d406_badpixel.fits
20:44:13.0 - * |-c:|Updating Calib Data Base with DARK
20:44:13.0 - * |-c:|Recipe -c has been succesfully completed
```

In interactive mode three figures will also appear (See figures 6.1, 6.2, and 6.3).

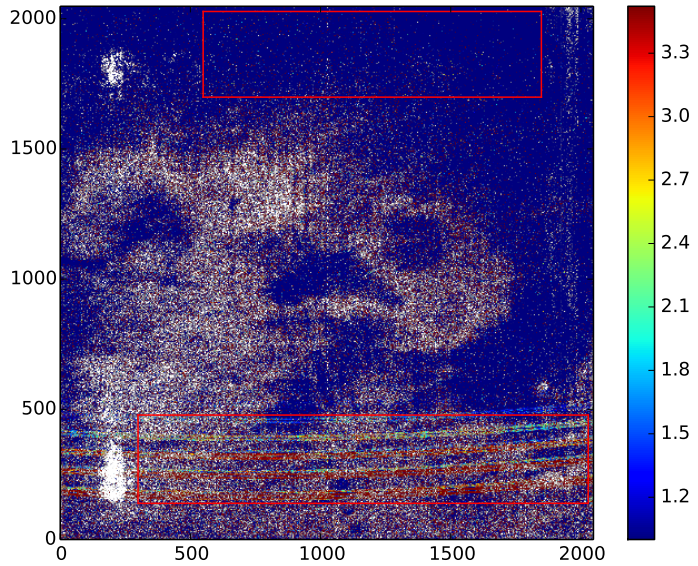

Figure 6.1

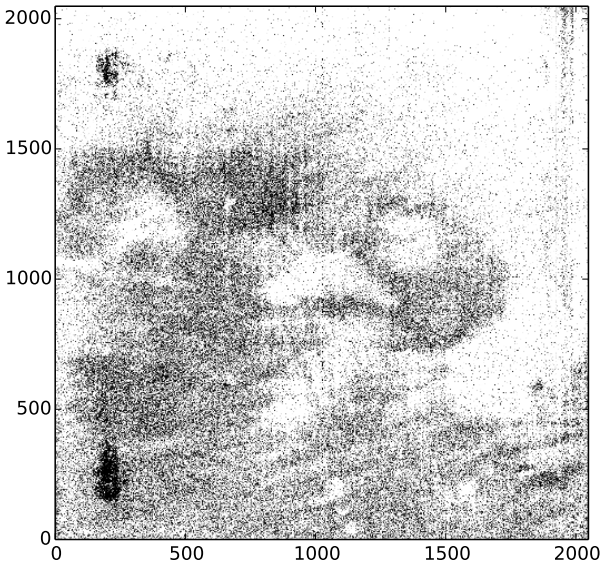

Figure 6.2

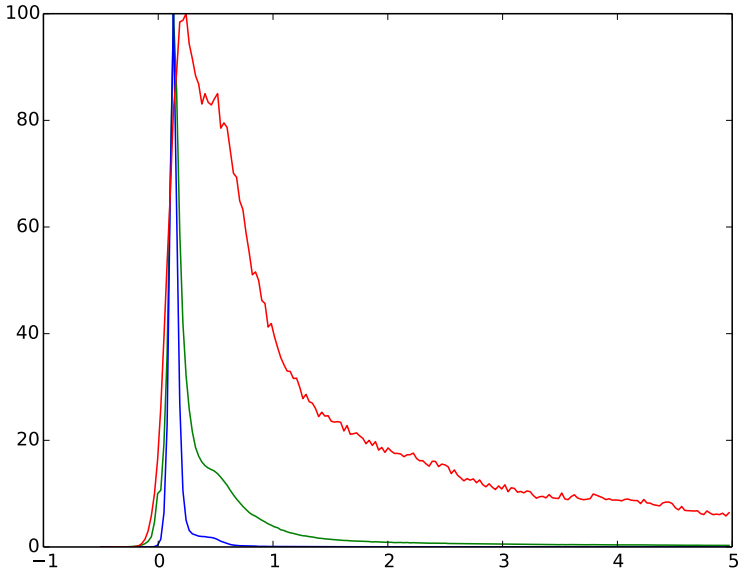

Figure 6.3

6.2.2 cal_loc_RAW_spirou

Some text here

6.2.3 cal_SLIT_spirou

Some text here

6.2.4 cal_SLIT_spirou

Some text here

6.2.5 cal_FF_RAW_spirou

Some text here

6.2.6 cal_extract_RAW_spirou

Some text here

6.2.7 cal_DRIFT_RAW_spirou

Some text here

6.3 Currently unused

6.3.1 cal_loc_ONE_spirou

Some text here

6.3.2 cal_FF_spirou

Some text here

6.3.3 cal_WAVE_spirou

Some text here

6.3.4 cal_FP_spirou

Some text here

6.3.5 cal_FF_spirou.py

Some text here

Chapter 7

Quality Control

Some text here

Appendix A

Source code for environmental setup file

```
#!/bin/bash
# activate using:
#    source env_setup.sh
# Use --clean to restore environment
# i.e.
#    source env_setup.sh --clean

# ------------------------------------------------------------------
# EDIT THESE PARAMETERS
# ------------------------------------------------------------------
# Set the instrument name
INSTRUMENT_NAME="SPIROU"
# Directory in which all SPIROU files go
DIR="/scratch/Projects/SPIRou_Pipeline"
# Define installation folder name
INSTALL_ROOT="$DIR/INTROOT"
# Define data folder name
DATA_ROOT="$DIR/data"
# Define raw path
DATA_ROOT_RAW="$DATA_ROOT/raw/"
# Define reduced path
DATA_ROOT_REDUCED="$DATA_ROOT/reduced/"
# Define calibDB path
DATA_ROOT_CALIB="$DATA_ROOT/calibDB"
# Define msg path
DATA_ROOT_MSG="$DATA_ROOT/msg/"
# Define tmp path
DATA_ROOT_TMP="$DATA_ROOT/tmp/"
# Define python version
PYTHON_VERSION="2.7"
# Define python directory (i.e. result of command "which python")
PYTHON_DIR="/scratch/Projects/SPIRou_Pipeline/miniconda2"
# Define GSL path
GSL_DIR="/scratch/Projects/SPIRou_Pipeline/c-libraries/gsl"

# ------------------------------------------------------------------
# DO NOT EDIT PAST THIS POINT
# ------------------------------------------------------------------
echo " ======================================== "
echo "  Environmental setup for DRS Pipeline    "
echo " ======================================== "

# get arguments
CLEAN="0"
for i in "$@"; do
```

```
  case $i in
    --clean) CLEAN="1";;
    *) echo "Invalid argument"; exit 1;;
  esac
done

if [ "$CLEAN" = "1" ]; then
  if [[ -z "${DRS_ACTIVE}" ]]; then
    if [ "$DRS_ACTIVE" != 1 ]; then
      echo "   No need to clean - environment clean"
      echo ""
      echo "   Done"
      exit
    fi
  fi
  echo "   Cleaning environment"
  export PATH=$OLDPATH
  export PYTHONPATH=$OLDPYTHONPATH
  unset OLDPATH
  unset DRS_ACTIVE
  unset OLDPYTHONPATH
  unset INTROOT
  unset DRS_LOG
  unset PYTHON_INCLUDE_DIR
  unset GSL_INCLUDE_DIR
  unset GSL_LIBRARY_DIR
  unset INSTRUMENT
  unset DRS_DATA_RAW
  unset DRS_DATA_REDUC
  unset DRS_CALIB_DB
  unset DRS_DATA_MSG
  unset DRS_DATA_WORKING
  unset TDATA
else
  echo "   Setting up environment"
  # Backup old path and python path for clean
  OLDPATH=$PATH
  OLDPYTHONPATH=$PYTHONPATH
  export OLDPATH
  export DRS_ACTIVE=1
  export OLDPYTHONPATH

  # User specific environment and startup programs
  export INTROOT="$INSTALL_ROOT"
  export PATH="$INSTALL_ROOT/bin":"$PYTHON_DIR/bin/":$PATH
  export PYTHONPATH=.:"$PYTHON_DIR/lib/python$PYTHON_VERSION/site-packages/":"$INSTALL_ROOT/bin"
  export DRS_LOG=1
  export PYTHON_INCLUDE_DIR="$PYTHON_DIR/lib/python$PYTHON_VERSION/site-packages/numpy/core/
     include"
  export GSL_INCLUDE_DIR="$GSL_DIR/include"
  export GSL_LIBRARY_DIR="$GSL_DIR/lib"

  #SPIROU DRS and DATA environment variables
  export INSTRUMENT=$INSTRUMENT_NAME
  export DRS_DATA_RAW=$DATA_ROOT_RAW
  export DRS_DATA_REDUC=$DATA_ROOT_REDUCED
  export DRS_CALIB_DB=$DATA_ROOT_CALIB
  export DRS_DATA_MSG=$DATA_ROOT_MSG
  export DRS_DATA_WORKING=$DATA_ROOT_TMP
  export TDATA=$DATA_ROOT

  echo "   Set up for $INSTRUMENT"
  echo "    - Python located at: $PYTHON_DIR"
```

```
  echo "    - GLS located at: $GSL_DIR"
  echo "    - installation located at: $INSTALL_ROOT"
  echo "    - data located at:"
  echo "           $DATA_ROOT"
  echo "           $DATA_ROOT_RAW"
  echo "           $DATA_ROOT_REDUCED"
  echo "           $DATA_ROOT_CALIB"
  echo "           $DATA_ROOT_MSG"
  echo "           $DATA_ROOT_TMP"
  echo " "

fi

echo "   Done"
```