# MEGARA Pipeline Documentation

***Release 0.8.dev0***

**Sergio Pascual**

**Dec 05, 2018**

# Contents

CHAPTER 1

# Overview

This guide is intended as an introductory overview of MEGARA Reduction Pipeline packages and explains how to install and make use of its most important features.

The MEGARA Pipeline can be used as an stand-alone application (referred as MEGARA Data Reduction Pipeline or MEGARA DRP) or integrated in the GTC Control System (MEGARA Data Factory Pipeline or MEGARA DFP).

For detailed reference documentation of the functions and classes contained in the package, see the *Reference*.

**Warning:** This "User Guide" is still a work in progress; some of the material is not organized, and several aspects of the MEGARA Pipeline are not yet covered in sufficient detail.

# CHAPTER 2

# Installation

## 2.1 Requirements

The MEGARA Pipeline package requires the following packages installed in order to be able to be installed and work properly:

- python[1] either 2.7 or >= 3.4
- setuptools[2]
- numpy[3] >= 1.7
- scipy[4]
- astropy[5] >= 2.0
- numina[6] >= 0.17
- scikit-image[7]

Additional packages are optionally required:

- py.test[8] >= 2.5 to run the tests
- sphinx[9] to build the documentation

## 2.2 Installing MEGARA DRP

### 2.2.1 Using Conda

*megaradrp* can be installed with conda using a custom channel.

[1] https://www.python.org
[2] http://peak.telecommunity.com/DevCenter/setuptools
[3] http://www.numpy.org/
[4] http://www.scipy.org/
[5] http://www.astropy.org/
[6] https://pypi.python.org/pypi/numina/
[7] http://scikit-image.org/
[8] http://pytest.org
[9] http://sphinx.pocoo.org

From the shell, execute::

```
conda install -c conda-forge megaradrp
```

### 2.2.2 Using pip

To install with pip, simply run::

```
pip install --no-deps megaradrp
```

**Note:** The `--no-deps` flag is optional, but highly recommended if you already have Numpy installed, since otherwise pip will sometimes try to upgrade your Numpy installation, which may not always be desired.

### 2.2.3 Building from source

The latest stable version of MEGARA DRP can be downloaded from https://pypi.python.org/pypi/megaradrp

To install MEGARA DRP, use the standard installation procedure:

```
$ tar zxvf megaradrp-X.Y.Z.tar.gz
$ cd megaradrp-X.Y.Z
$ python setup.py install
```

The *install* command provides options to change the target directory. By default installation requires administrative privileges. The different installation options can be checked with:

```
$ python setup.py install --help
```

### 2.2.4 Checking the installation

Once the installation is finished, you can check by listing the installed recipes with the command line interface tool `numina`:

```
(myenv) $ ./bin/numina show-instruments
INFO: Numina simple recipe runner version 0.13.0
Instrument: MEGARA
 has configuration 'default'
 has pipeline 'default', version 1
 has pipeline 'experimental', version 1
```

#### Development version

The development version can be checked out with:

```
$ git clone https://github.com/guaix-ucm/megaradrp.git
```

And then installed following the standard procedure:

```
$ cd megaradrp
$ python setup.py install
```

#### Building the documentation

The MEGARA DRP documentation is base on sphinx[10]. With the package installed, the html documentation can be built from the *doc* directory:

```
$ cd doc
$ make html
```

The documentation will be copied to a directory under *build/sphinx*.

The documentation can be built in different formats. The complete list will appear if you type *make*

### 2.2.5 Deployment with Virtualenv

Virtualenv[11] is a tool to build isolated Python environments.

It's a great way to quickly test new libraries without cluttering your global site-packages or run multiple projects on the same machine which depend on a particular library but not the same version of the library.

#### Install virtualenv

I install it with the package system of my OS, so that it ends in my global site-packages.

With Fedora/EL is just:

```
$ sudo yum install python-virtualenv
```

#### Create virtual environment

Create the virtual environment enabling the packages already installed in the global site-packages via the OS package system. Some requirements (in particullar numpy and scipy) are difficult to build: they require compiling and external C and FORTRAN libraries to be installed.

So the command is:

```
$ virtualenv --system-site-packages myenv
```

If you need to create the virtualenv without global packages, drop the system-site-packages flag.

#### Activate the environment

Once the environment is created, you need to activate it. Just change directory into it and load with your command line interpreter the script bin/activate.

With bash:

```
$ cd myenv
$ . bin/activate
(myenv) $
```

With csh/tcsh:

```
$ cd myenv
$ source bin/activate
(myenv) $
```

[10] http://sphinx.pocoo.org
[11] http://pypi.python.org/pypi/virtualenv

Notice that the prompt changes once you are activate the environment. To deactivate it just type deactivate:

```
(myenv) $ deactivate
$
```

## 2.3 Installing MEGARA DFP

This section described how to install the MEGARA Pipeline inside the GTC Control system.

In the following we assume that we are installing with user *gcsop*.

Login in the *gcsop* account and activate the GTC environment:

```
$ /opt/gcs/tools/nrp -p linux -s bash
```

Change working directory to `/work/gcsop/src_python/gtc`:

```
$ cd /work/gcsop/src_python/gtc
$ ls
AL  DSL  SSL
```

We have to install *numina* under *DSL* and *megaradrp* under *AL*.

Please refer to Numina manual[12] to install Numina and its dependences under Solaris 10.

### 2.3.1 Install numina

First, install all the dependencies:

- setuptools
- six
- numpy >= 1.7
- scipy
- astropy >= 1.0
- PyYaml
- singledispatch

If you are installing a development version, Cython is also required.

Most are available as precompiled packages in Linux. Please refer to Numina manual[13] to install Numina and its dependences under Solaris 10.

Then, download the source code, either from PyPI or github:

```
$ pwd
/work/gcsop/src_python/gtc/DSL/
$ git clone https://github.com/guaix-ucm/numina.git
$ cd numina
```

Create a file *numina.mod* with the following content:

[12] https://numina.readthedocs.io/en/latest/user/solaris.html#solaris10
[13] https://numina.readthedocs.io/en/latest/user/solaris.html#solaris10

```
NAME=numina
TYPE=device

l:numina:python:y
```

And then build and install using *nmk*:

```
$ nmk -t module.rebuild
$ nmk -t module.install
```

### 2.3.2 Install megaradrp

Change directory to */work/gcsop/src_python/gtc/AL/* and download the source code of *megaradrp*, either from PyPI[14] or from github[15]:

```
$ pwd
/work/gcsop/src_python/gtc/AL/
$ git clone https://github.com/guaix-ucm/megaradrp.git
$ cd megaradrp
```

Create a file *megaradrp.mod* with the following content:

```
NAME=megaradrp
TYPE=device

l:megaradrp:python:y
```

And then build and install using *nmk*:

```
$ nmk -t module.rebuild
$ nmk -t module.install
```

You can check that everything works by running the *numina* command line tool:

```
$ numina show-instruments
Instrument: MEGARA
 has configuration 'default'
 has pipeline 'default', version 1
```

[14] https://pypi.python.org/pypi/megaradrp
[15] https://github.com/guaix-ucm/megaradrp

# CHAPTER 3

# Testing

This section describes the testing framework and options for testing MEGARA DRP

## 3.1 Running tests

MEGARA DRP uses py.test[16] as its testing framework.

As MEGARA DRP does not contain C/Cython extensions, the tests can be run directly in the source code, as:

```
cd megaradrp-0.4.0
cd src
py.test megaradrp
```

Some of the tests rely on data downloaded from a server. These tests are skipped by default. To enable them run instead:

```
py.test --run-remote megaradrp
```

The reduction recipes are tested with remote data. Each recipe is run in a directory created under the default `$TMPDIR`, which is based on the user temporal directory. The base of the created directories can be changed with the option `--basetemp=dir`:

```
py.test --basetemp=/home/spr/test100 --run-remote megaradrp
```

[16] http://pytest.org

CHAPTER 4

# Running the pipeline

The MEGARA DRP is run through a command line interface provided by **numina**.

The run mode of numina requires:

- A observation result file in YAML[17] format
- A requirements file in YAML format
- The raw images obtained in the observing block
- The calibrations required by the recipe

The observation result file and the requirements file are created by the user, the format is described in the following sections.

## 4.1 Format of the observation result

The contents of the file is a serialized dictionary with the following keys:

***id*: not required, string, defaults to 1** Unique identifier of the observing block

***instrument*: required, string** Name of the instrument, as it is returned by `numina show-instruments`

***mode*: required, string** Name of the observing mode, as returned by `numina show-modes`

***frames*: required, list of strings** List of images names

***children*: not required, list of integers, defaults to empty list** Identifications of nested observing blocks

This is an example of the observation result file

```
id: dark-test-21
instrument: MEGARA
mode: MegaraDarkImage
images:
  - r0121.fits
  - r0122.fits
```

(continues on next page)

17 http://www.yaml.org

(continued from previous page)

```
  - r0123.fits
  - r0124.fits
  - r0125.fits
  - r0126.fits
  - r0127.fits
  - r0128.fits
  - r0129.fits
  - r0130.fits
  - r0131.fits
  - r0132.fits
```

## 4.2 Format of the requirements file

This file contains calibrations obtained by running recipes (called **products**) and other parameters (numeric or otherwise) required by the recipes (named **requirements**). The file is serialized using YAML[18]

Example requirements file:

```
version: 1                                                       (1)
products:                                                        (2)
  EMIR:
   - {id: 1, content: 'file1.fits', type: 'MasterFlat', tags: {'filter': 'J'}, ob:␣
↪200}      (3)
   - {id: 4, content: 'file4.fits', type: 'MasterBias', tags: {'readmode': 'cds'},␣
↪ob: 400} (3)
  MEGARA:
   - {id: 1, content: 'file1.fits', type: 'MasterFiberFlat', tags: {'vph': 'LR-U'},
↪ ob: 1200} (3)
   - {id: 2, content: 'file2.yml', type: 'TraceMap', tags: {'vph': 'LR2', 'readmode
↪': 'fast'}, ob: 1203} (3)
requirements: (4)
  MEGARA:
     default:
       MegaraArcImage:  (5)
          polynomial_degree: 5 (6)
          nlines: [5, 5]       (6)
```

1. Mandatory entry, `version` must be 1
2. Products of other recipes are list, by instrument
3. **The products of the reduction recipes are listed. Each result must contain:**
   - A `type`, one of the types of the products of the DRP in string format
   - A `tags` field, used to select the correct calibration based on the keywords of the input.
   - A `content` field, a pointer to the serialized version of the calibration.
   - A `id` field, unique integer
   - A `ob` field, optional integer, used to store the observation id of the images that created the calibration.
4. Numerical parameters of the recipes are stored in `requirements`, with different sections per instrument.
5. The name of the observing mode.
6. Different parameters for the recipe corresponding to the observing mode in (5)

[18] http://www.yaml.org

## 4.3 Running the pipeline

**numina** copies the images (calibrations and raw data) from directory `datadir` to directory `workdir`, where the processing happens. The result is stored in directory `resultsdir`. The default values are for each directory are `data`, `obsid<id_of_obs>_work` and `obsid<id_of_obs>_results`. All these directories can be defined in the command line using flags:

```
$ numina run --workdir /tmp/test1 --datadir /scrat/obs/run12222 obs.yaml -r␣
↪requires.yaml
```

See Command Line Interface[19] for a full description of the command line interface.

Following the example, we create a directory `data` in our current directory and copy there the raw frames from `r0121.fits` to `r0132.fits` and the master bias `master_bias-1.fits`.

The we run:

```
$ numina run obsresult.yaml -r requirements.yaml
INFO: Numina simple recipe runner version 0.15
INFO: Loading observation result from 'obsrun.yaml'
INFO: Identifier of the observation result: 1
INFO: instrument name: MEGARA
...
numina.recipes.megara INFO stacking 4 images using median
numina.recipes.megara INFO bias reduction ended
INFO: result: BiasRecipeResult(qc=Product(type=QualityControlProduct(), dest='qc'),
↪ biasframe=Product(type=MasterBias(), dest='biasframe'))
INFO: storing result
```

We get information of what's going on through logging messages. In the end, the result and log files are stored in `obsid<id_of_obs>_results`. The working directory `obsid<id_of_obs>_work` can be inspected too. Intermediate results will be saved here.

On the other hand, in the following we attach a short code to run megaradrp by using a Python script. This is useful to use the Python debugger.

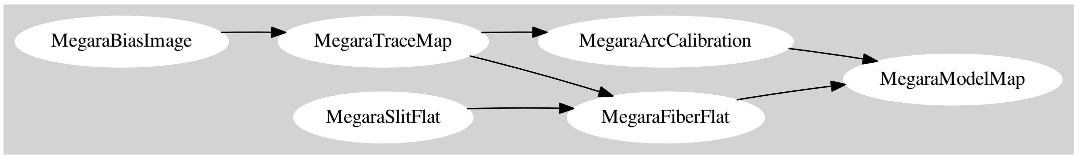
```python
from numina.user.cli import main
from megaradrp.loader import load_drp

def run_recipe():
    main(['run', 'obsresult.yaml', '-r', 'requirements.yaml'])

if __name__ == "__main__":
    run_recipe()
```

### 4.3.1 Pipeline's Flow Example

In this subsection, we detail an example about how to generate a called Master Fiber Flat Image. To achieve our goal, a schematic flow can be seen in the next Figure:



[19] https://numina.readthedocs.io/en/latest/user/cli.html#cli

It is important to emphasize the fact that each time a Recipe is run, the results must be renamed and copied to the `data` directory in order to be the input of the next Recipe if it is needed. Taking this in mind, the content of the `requirements.yaml` file might well be and is common to all Recipes:

```
version: 1
products:
  MEGARA:
  - {id: 1, type: 'LinesCatalog', tags: {}, content: 'ThAr_arc_LR-U.txt'}
  - {id: 2, type: 'MasterBias', tags: {}, content: 'master_bias.fits'}
  - {id: 3, type: 'TraceMap', tags: {}, content: 'master_traces.json'}
  - {id: 4, type: 'MasterFiberFlat', tags: {}, content: 'master_fiberflat.fits'}
  - {id: 5, type: 'WavelengthCalibration', tags: {}, content: 'master_wlcalib.json
↪'}
  - {id: 6, type: 'MasterFiberFlatFrame', tags: {}, content: 'fiberflat_frame.fits
↪'}
  - {id: 7, type: 'ModelMap', tags: {}, content: 'master_model.json'}
  - {id: 8, type: 'MasterSlitFlat', tags: {}, content: 'master_slitflat.fits'}
requirements: {}
```

In order to run the next example, the user should execute the next command at least 6 times taking into account that the file `obsresult-%step.yaml` should change with each execution:

```
$ numina run obsresult-1.yaml -r requirements.yaml
$ numina run obsresult-2.yaml -r requirements.yaml
...
$ numina run obsresult-6.yaml -r requirements.yaml
```

MegaraBiasImage file, obsresult-1.yaml:

```
id: 1
instrument: MEGARA
mode: MegaraBiasImage
images:
  - bias1.fits
  - bias2.fits
  - bias3.fits
  - bias4.fits
  - bias5.fits
```

MegaraTraceMap, obsresult-2.yaml:

```
id: 2
instrument: MEGARA
mode: MegaraTraceMap
images:
  - flat1.fits
  - flat2.fits
  - flat3.fits
  - flat4.fits
  - flat5.fits
```

MegaraArcCalibration, obsresult-3.yaml:

```
id: 3
instrument: MEGARA
mode: MegaraArcCalibration
images:
  - arc1.fits
  - arc2.fits
  - arc3.fits
  - arc4.fits
  - arc5.fits
```

MegaraSlitFlat, obsresult-4.yaml:

```
id: 4
instrument: MEGARA
mode: MegaraSlitFlat
images:
  - flat1.fits
  - flat2.fits
  - flat3.fits
  - flat4.fits
  - flat5.fits
```

MegaraModel, obsresult-5.yaml:

```
id: 5
instrument: MEGARA
mode: MegaraModelMap
images:
  - flat1.fits
  - flat2.fits
  - flat3.fits
  - flat4.fits
  - flat5.fits
```

MegaraFiberFlat, obsresult-6.yaml:

```
id: 6
instrument: MEGARA
mode: MegaraFiberFlat
images:
  - flat1.fits
  - flat2.fits
  - flat3.fits
  - flat4.fits
  - flat5.fits
```

Notice that if you would want to execute this example automatically, you could code a script (following the same skeleton as shown above) with a loop flow to read the .yaml files and the outputs that each recipe generates.

CHAPTER 5

# Observing modes

Observing modes are prescribed method of observing with MEGARA. Each observing mode is processed with one reduction recipe. In the following we describe the observing modes and its corresponding recipe. The **Usage** field describes if the mode is to be used in the telescope (*Online*, DFP Mode) or after the observation by the user (*Offline*, DRP Mode).

Observing modes are described in full detail elsewhere (document TEC/MEG/05)

## 5.1 Calibration Modes

The calibration modes are those operating modes that are intended (1) to either analyze the state of the instrument or (2) to be used for processing scientific observations from raw to science-grade data.

With respect to the determination of the status of the instrument the following calibration images should be acquired:

- Bias
- Dark
- Fiber-flat
- Arc

Regarding the processing of scientific data this basically implies obtaining calibration images in number and quality required to remove the instrumental signatures so to obtain a science-grade image. These images, which are taken as part of routine scientific operations, include:

- Bias
- Dark
- Slit-flat
- Fiber-flat
- Twilight fiber-flat
- Arc
- Standard star

Except for the slit-flat, that might be taken only occasionally, the rest of this latter set of observing modes will be taken routinely as part of either daytime or nighttime operations. We will refer to these as "Daily CalibrationModes". Besides these modes we have identified a series of calibration modes (named "System Calibration Modes") that are also necessary for processing MEGARA observations but that are only produced occasionally as part of long-term calibrations of the instrument to be carried out by the observatory staff.

Thus, the "System Calibration Modes" will be:

- Bad-pixels mask
- Linearity Test
- Slit-flat. Whether the slit-flat should be considered as a "Daily Calibration" or "System Calibration" mode is TBD and will depend on the stability of the pixel-to-pixel efficiency of the MEGARA CCD.

In this latter case, the difference between a "System Calibration Mode" and the corresponding "Auxiliary Mode" described in Section 3 depends on the frequency the observing mode has to be executed. Auxiliary modes are typically run once every observing run (e.g. the fine-acquisition ones) or, in the best case, after a long period of inactivity. System Calibration modes, on the other hand, are expected to be run only after major changes in the telescope or the instrument or if a degradation of any of the subsystems of the instrument is suspected.

The need for obtaining all these sets of images drives the requirements and characteristics of the Calibration modes described below as defined by the MEGARA team.

## 5.1.1 Bias Image

**Mode** Bias

**Usage** Offline, Online

**Key** MegaraBIAS_IMAGE

**Product** `MasterBias`

**Recipe** `BiasRecipe`

**Recipe input** `RecipeInput`

**Recipe result** `RecipeResult`

Before the Analog-to-Digital conversion is performed a pedestal (electronic) level is added to all images obtained with the MEGARA CCD. This is a standard procedure in CCD imaging and spectroscopy applications for Astronomy and is intended to minimize the ADC errors produced when very low analog values are converted to DUs.

### Requirements of the mode

The sequence for this observing mode should include the actions to calibrate the pedestal level of the detectors and associated control electronics by taking images with null integration time. This mode requires having the shutter closed and to readout the detector in a series of exposures with null integration time, being this series the bias image set.

### Procedure

The frames in the observed block are stacked together using the median of them as the final result. The variance of the result frame is computed using two different methods. The first method computes the variance across the pixels in the different frames stacked. The second method computes the variance in each channel in the result frame.

### Products

Bias image sets are to be obtained both as part of the activities related to the verification of the instrument status and for processing data for scientific exploitation.

### Recipe, inputs and results

**class** `megaradrp.recipes.calibration.bias.`**`BiasRecipe`**(**args*, ***kwargs*)

Process BIAS images and create a MASTER_BIAS product.

This recipe process a set of bias images obtained in **Bias Image** mode and returns a combined product image, trimmed to the physical size of the detector.

**See also:**

*`megaradrp.types.MasterBias`* description of the MasterBias product

#### Notes

Images are corrected from overscan and trimmed to the physical size of the detector. Then, they corrected from Bad Pixel Mask, if the BPM is available, Finally, images are stacked using the median.

**class** **`BiasRecipeInput`**(**args*, ***kwds*)

BiasRecipeInput documentation.

**Attributes**

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**obresult** [ObservationResultType, requirement] Observation Result

**class** **`BiasRecipeResult`**(**args*, ***kwds*)

BiasRecipeResult documentation.

**Attributes**

**master_bias** [MasterBias, product]

**qc** [QualityControlProduct, product]

**`RecipeInput`**

alias of `numina.core.metaclass.BiasRecipeInput`

**`RecipeResult`**

alias of `numina.core.metaclass.BiasRecipeResult`

**`run`**(*rinput*)

Execute the recipe.

**Parameters**

**rinput** [BiasRecipe.RecipeInput]

**Returns**

**BiasRecipe.RecipeResult**

**`set_base_headers`**(*hdr*)

Set metadata in FITS headers.

### 5.1.2 Dark Image

**Mode** Dark

**Usage** Offline, Online

**Key** MegaraDarkImage

**Product** *MasterDark*

**Recipe** *DarkRecipe*

**Recipe input** `DarkRecipeInput`

**Recipe result** `DarkRecipeResult`

The potential wells in CCD detectors spontaneously generate electron-ion pairs at a rate that is a function of temperature. For very long exposures this translates into a current that is associated with no light source and that is commonly referred to as dark current.

#### Requirements

While in imaging or low-resolution spectroscopy this is nowadays a negligible effect thanks to the extremely low dark current levels of state-of-the-art CCDs (typically < 1 e-/hour) when working at intermediate-to-high spectral resolutions where the emission per pixel coming from the sky background and the astronomical source can be very low this is worth considering.

The sequence for this observing mode should include the actions to measure the variation of the intrinsic signal of the system by taking images under zero illumination condition and long integration time. This mode requires that the focal-plane cover is configured (it should be fully closed), the shutter is closed and to expose a certain time and readout the detector a series of exposures, being this series the dark image set.

#### Procedure

The "User" processes an observing block obtained in the observing mode Dark. This mode includes the required actions to obtain a master dark frame. The master dark generated is used in other stages of the data processing.

#### Products

Dark image sets are to be obtained both as part of the activities related to the verification of the instrument status and for processing data for scientific exploitation.

A bidimensional dark image, QA flag, a text log file of the processing and a structured text file containing information about the processing.

#### Recipe, inputs and results

**class** `megaradrp.recipes.calibration.dark.`**`DarkRecipe`**(**args*, ***kwargs*)

Process DARK images and provide MASTER_DARK.

**class** **`DarkRecipeInput`**(**args*, ***kwds*)

DarkRecipeInput documentation.

**Attributes**

**master_bias** [MasterBias, requirement] Master BIAS image

**obresult** [ObservationResultType, requirement] Observation Result

**class DarkRecipeResult**(**args, **kwds*)
DarkRecipeResult documentation.

**Attributes**

**master_dark** [MasterDark, product]

**qc** [QualityControlProduct, product]

**RecipeInput**
alias of `numina.core.metaclass.DarkRecipeInput`

**RecipeResult**
alias of `numina.core.metaclass.DarkRecipeResult`

**set_base_headers**(*hdr*)
Set metadata in FITS headers.

### 5.1.3 Slit-flat

**Mode** Slit-flat

**Usage** Offline, Online

**Key** MegaraSlitFlat

**Product** *MasterSlitFlat*

**Recipe** *SlitFlatRecipe*

**Recipe input** `SlitFlatRecipeInput`

**Recipe result** `SlitFlatRecipeResult`

In the case of fiber-fed spectrographs the correction for the detector pixel-to-pixel variation of the sensibility is usually carried out using data from laboratory, where the change in efficiency of the detector at different wavelengths is computed and then used to correct for this effect for each specific instrument configuration (VPH setup in the case of MEGARA).

#### Requeriments

In the case of MEGARA we will offset the pseudo-slit from its optical focus position to ensure that the gaps between fibers are also illuminated when a continuum (halogen) lamp at the ICM is used. The NSC zemax model of the spectrograph indicates that by offsetting 3mm the pseudo-slit we would already obtain a homogenous illumination of the CCD. A series of images with different count levels would be obtained.

The quality of present-day CCDs leads to a rather small impact of these pixel-to-pixel variations in sensitivity on either the flux calibration and the cosmetics of the scientific images, especially considering that not one but a number of pixels along the spatial direction are extracted for each fiber and at each wavelength. Therefore, we anticipate that this correction might not be needed or that, as a maximum, a first-order correction based on laboratory data might suffice. However, before the results of the analysis of the pixel-to-pixel variations in sensitivity planned using our CCD230 e2V test CCD are obtained we will consider this observing mode as TBC.

This mode requires having the ICM halogen lamp on, the instrument shutter open, to move the pseudo-slit to the open position, to configure the VPH wheel mechanism in order to select the grating to be used, to move the focusing mechanism to the position pre-defined for the specific VPH of choice but offset by 3mm and to expose a certain time and to readout the detector a series of exposures, being this series the slit-flat image set.

### Procedure

The "User" processes an observing block obtained in the observing mode Slit-flat. This mode includes the required actions to obtain a master slit-flat field. The master slit-flat field generated is used in other stages of the data processing.

### Products

Slit-flat image sets are to be obtained both as part of the activities related to the verification of the instrument status (such as for evaluating the status of the MEGARA spectrograph) and also for processing data for scientific exploitation (correction for the pixel-to-pixel variation in sensitivity). The frequency at which these detector flat images should be acquired is TBC. Although defined in this document as a mode to be considered part of the "Daily Calibration Modes" if it is finally used only sporadic it should be considered as part of the "System Calibration Modes" instead.

A bidimensional master slit flat field, QA flag, a text log file of the processing and a structured text file containing information about the processing.

### Recipe, inputs and results

**class** `megaradrp.recipes.calibration.slitflat.`**`SlitFlatRecipe`**(**args*, ***kwargs*)
Process SLIT_FLAT images and create MasterSlitFlat.

**`RecipeInput`**
alias of `numina.core.metaclass.SlitFlatRecipeInput`

**`RecipeResult`**
alias of `numina.core.metaclass.SlitFlatRecipeResult`

**class `SlitFlatRecipeInput`**(**args*, ***kwds*)
SlitFlatRecipeInput documentation.

**Attributes**

**master_bias** [MasterBias, requirement] Master BIAS image

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**master_dark** [MasterDark, requirement, optional] Master DARK image

**obresult** [ObservationResultType, requirement] Observation Result

**class `SlitFlatRecipeResult`**(**args*, ***kwds*)
SlitFlatRecipeResult documentation.

**Attributes**

**master_slitflat** [MasterSlitFlat, product]

**qc** [QualityControlProduct, product]

**reduced_image** [ProcessedFrame, product]

**`set_base_headers`**(*hdr*)
Set metadata in FITS headers.

## 5.1.4 Trace

**Mode** Trace

**Usage** Offline, Online

**Key** MegaraTraceMap

**Product** *`TraceMap`*.

**Recipe** *TraceMapRecipe*

**Recipe input** `TraceMapRecipeInput`

**Recipe result** `TraceMapRecipeResult`

Although for the majority of the observing modes described elsewhere in this document the MEGARA off-line pipeline will perform its own fiber spectra extraction from the 2D CCD FITS frame, there are cases where an archival master "trace map" should be used instead. Note that a different "trace map" should be available for each pseudo-slit and VPH combination.

#### Requirements

This observing mode should include the actions needed to obtain a series of Fiber-flats that should be combined to generate a master "trace map". This will be done by means of illuminating the instrument focal plane with a continuum (halogen) lamp that is part of the GTC Instrument Calibration Module (ICM). The use of the twilight sky is not recommended in this case as the twilight sky can present strong absorption lines that could lead to errors in the resulting trace map at specific wavelengths.

This mode requires having the ICM turned on, one of the halogen lamps at the ICM also turned on, to configure the focal-plane cover (at least one of the sides should be open), to have the instrument shutter open, to move the pseudo-slit to that of the instrument mode of choice, to configure the VPH wheel mechanism in order to select the grating to be used, to move the focusing mechanism to the position pre-defined for the specific VPH of choice and to expose a certain time and to readout the detector a series of exposures, being this series the trace map image set.

#### Procedure

The "User" processes an observing block obtained in the observing mode Trace. This mode includes the required actions to obtain a mapping of the trace of the fibers. The master trace map generated is used in other stages of the data processing.

#### Products

Trace map image sets are to be obtained both as part of the activities related to the verification of the instrument status and for processing data for scientific exploitation. Note, however, that the use of this observing mode for scientific exploitation should be limited as it could affect to the general performance of the on-line quick-look software.

This mode produces the tracing information required to extract the flux of the fibers. The result is stored in an object named `master_traces` of type *TraceMap*.

#### Recipe

**class** `megaradrp.recipes.calibration.trace.`**`TraceMapRecipe`**(**args*, ***kwargs*)

Provides tracing information from continuum flat images.

This recipe process a set of continuum flat images obtained in *Trace Map* mode and returns the tracing information required to perform fiber extraction in other recipes. The recipe also returns the result of processing the input images upto dark correction.

**See also:**

***`megaradrp.products.tracemap.TraceMap`*** description of TraceMap product

***`megaradrp.recipes.calibration.modelmap.ModelMapRecipe`*** description of ModelMap recipe

**`numina.array.trace.traces`** tracing algorithm `megaradrp.instrument.configs` instrument configuration

### Notes

Images provided in *obresult* are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is not None), bias and dark current (if *master_dark* is not None). Images thus corrected are the stacked using the median.

The result of the combination is saved as an intermediate result, named 'reduced_image.fits'. This combined image is also returned in the field *reduced_image* of the recipe result and will be used for tracing the position of the fibers.

The fibers are grouped in packs of different numbers of fibers. To match the traces in the image with the corresponding fibers is neccessary to know how fibers are packed and where the different groups of fibers appear in the detector. This information is provided by the fields 'pseudoslit.boxes' and 'pseudoslit.boxes_positions' of the instrument configuration.

Using the column reference provided by 'pseudoslit.boxes_positions', peaks are detected (using an average of 7 columns) and matched to the layout of fibers provided by 'pseudoslit.boxes_positions'. Fibers without a matching peak are counted and their ids stored in the final *master_traces* object.

Once the peaks in the reference column are found, each one is traced until the border of the image is reached. The trace may be lost before reaching the border. In all cases, the beginning and the end of the trace are stored.

The Y position of the trace is fitted to a polynomial of degree *polynomial_degree*. The coefficients of the polynomial are stored in the final *master_traces* object.

**RecipeInput**
alias of `numina.core.metaclass.TraceMapRecipeInput`

**RecipeResult**
alias of `numina.core.metaclass.TraceMapRecipeResult`

**class TraceMapRecipeInput**(**args*, ***kwds*)
TraceMapRecipeInput documentation.

**Attributes**

**debug_plot** [int, requirement, optional] Save intermediate tracing plots

**master_bias** [MasterBias, requirement] Master BIAS image

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**master_dark** [MasterDark, requirement, optional] Master DARK image

**obresult** [ObservationResultType, requirement] Observation Result

**polynomial_degree** [int, requirement, optional, default=5] Polynomial degree of trace fitting

**relative_threshold** [float, requirement, optional, default=0.3] Threshold for peak detection

**class TraceMapRecipeResult**(**args*, ***kwds*)
TraceMapRecipeResult documentation.

**Attributes**

**master_traces** [TraceMap, product]

**qc** [QualityControlProduct, product]

**reduced_image** [ProcessedImage, product]

**reduced_rss** [ProcessedRSS, product]

**refine_boxes_from_image** (*reduced*, *expected*, *cstart=2000*, *nsearch=20*)
Refine boxes using a filtered Fourier image

**run** (*rinput*)
Execute the recipe.

**Parameters**

**rinput** [TraceMapRecipe.RecipeInput]

**Returns**

**TraceMapRecipe.RecipeResult**

**Raises**

**ValidationError** if the recipe input is invalid

**run_qc** (*recipe_input*, *recipe_result*)
Run quality control checks

### 5.1.5 Model Map

**Mode** ModelMap

**Usage** Offline

**Key** MegaraModelMap

**Product** *ModelMap*.

**Recipe** *ModelMapRecipe*

**Recipe input** `ModelMapRecipeInput`

**Recipe result** `ModelMapRecipeResult`

Although for the majority of the observing modes described elsewhere in this document the MEGARA off-line pipeline will perform its own fiber spectra extraction from the 2D CCD FITS frame, there are cases where an archival master "model map" should be used instead. Note that a different "model map" should be available for each pseudo-slit and VPH combination.

#### Requirements

This offline observing mode will use the images obtained in the online observing mode "Trace".

#### Procedure

The "User" processes an observing block obtained in the observing mode Trace. This mode includes the required actions to obtain a mapping of the profiles of the fibers. The master model map generated is used in other stages of the data processing.

#### Products

Trace map image sets are to be obtained both as part of the activities related to the verification of the instrument status and for processing data for scientific exploitation. Note, however, that the use of this observing mode for scientific exploitation should be limited as it could affect to the general performance of the on-line quick-look software.

This mode produces the profile information required to perform advanced extraction of the fibers. The result is stored in an object named `master_model` of type *ModelMap*.

### Recipe

**class** `megaradrp.recipes.calibration.modelmap.`**`ModelMapRecipe`**(**args*, ***kwargs*)

Provides fiber profile information from continuum flat images.

This recipe process a set of continuum flat images obtained in *Trace Map* mode and returns the fiber profile information required to perform advanced fiber extraction in other recipes. The recipe also returns the result of processing the input images upto dark correction.

**See also:**

`megaradrp.products.modelmap.ModelMap` description of ModelMap product

**`megaradrp.recipes.calibration.tracemap.TraceMapRecipe`** description of TraceMap recipe

**`megaradrp.instrument.configs`** instrument configuration

#### Notes

Images provided in *obresult* are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is not None), bias and dark current (if *master_dark* is not None). Images thus corrected are the stacked using the median.

The result of the combination is saved as an intermediate result, named 'reduced_image.fits'. This combined image is also returned in the field *reduced_image* of the recipe result and will be used for fiting the profiles of the fibers.

The approximate central position of the fibers is obtained from *master_traces*. Then, for each 100 columns of the reduced image, a vertical cut in the image is fitted to a sum of fiber profiles, being the profile a gaussian convolved with a square.

The fits are made in parallel, being the number of processes controlled by the parameter *processes*, with the default value of 0 meaning to use the number of cores minus 2 if the number of cores is greater or equal to 4, one process otherwise.

After the columns are fitted, the profiles (central position and sigma) are interpolated to all columns using splines. The coefficientes of the splines are stored in the final *master_traces* object.

The recipe returns also the RSS obtained by applying advanced extraction to *reduced_image*. As an intermediate result, the recipe procedures DS9 reg files with the position of the center of the profiles, that can be used with raw and reduced images.

**class** **`ModelMapRecipeInput`**(**args*, ***kwds*)

ModelMapRecipeInput documentation.

**Attributes**

**debug_plot** [int, requirement, optional] Save intermediate tracing plots

**master_bias** [MasterBias, requirement] Master BIAS image

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**master_dark** [MasterDark, requirement, optional] Master DARK image

**master_slitflat** [MasterSlitFlat, requirement, optional] Master slit flat calibration

**master_traces** [TraceMap, requirement] Trace information of the Apertures

**obresult** [ObservationResultType, requirement] Observation Result

**processes** [int, requirement, optional] Number of processes used for fitting

**class** **`ModelMapRecipeResult`**(**args*, ***kwds*)

ModelMapRecipeResult documentation.

**Attributes**

**master_model** [ModelMap, product]

**qc** [QualityControlProduct, product]

**reduced_image** [ProcessedImage, product]

**reduced_rss** [ProcessedRSS, product]

**`RecipeInput`**
alias of `numina.core.metaclass.ModelMapRecipeInput`

**`RecipeResult`**
alias of `numina.core.metaclass.ModelMapRecipeResult`

### 5.1.6 Arc

**Mode** Arc

**Usage** Offline, Online

**Key** MegaraArcCalibration

**Product** *`WavelengthCalibration`*

**Recipe** *`ArcCalibrationRecipe`*

**Recipe input** `ArcCalibrationRecipeInput`

**Recipe result** `ArcCalibrationRecipeResult`

This mode sequence includes the required actions to translate the geometrical position of each point in the detector into physical units of wavelength. The calibration is performed by means of using reference calibration lamps (arc lamps) that should be part of the Instrument Calibration Module (ICM) at F-C. Note that the optical distortions in the spectrograph will lead to different wavelength calibrations from each individual fiber, therefore the entire focal plane should be illuminated by the corresponding arc lamp of choice. Given the relatively high spectral resolution and broad wavelength coverage of MEGARA we anticipate that more than one arc lamp will be needed at the ICM. The lamps at ICM have to deliver enough bright spectral lines for calibrating the whole range of MEGARA spectral resolutions and wavelength ranges (for HR modes only two VPHs shall be provided by MEGARA but more gratings could come funded by GTC users). MEGARA has provided a whole review of the possible illumination systems in the document TEC/MEG/151, but the responsibility of the development of the ICM module is on the GTC side.

#### Requirements

The entire focal plane should be illuminated with light from the ICM arc lamp with the required input focal ratio. This mode requires having the ICM turned on, one of the arc lamps at the ICM also turned on, the focal-plane cover configured (at least one of the sides should be open), the instrument shutter open, to move the pseudo-slit to that of the instrument mode of choice, to configure the VPH wheel mechanism in order to select the grating to be used, to move the focusing mechanism to the position pre-defined for the specific VPH of choice and to expose a certain time and to readout the detector a series of exposures, being this series the arc image set.

#### Procedure

The "User" processes an observing block obtained in the observing mode Arc. This mode includes the required actions to translate the geometrical position of each point in the detector into physical units of wavelength. The wavelength calibration generated is used in other stages of the data processing.

### Products

Arc image sets are to be obtained both as part of the activities related to the verification of the instrument status and for processing data for scientific exploitation and are part of the "Daily Calibration Modes".

A data structure containing information about wavelength calibrations (the format is TBD), a QA flag, a text log file of the processing and a structured text file containing information about the processing.

### Recipe, inputs and results

**class** `megaradrp.recipes.calibration.arc.`**`ArcCalibrationRecipe`**(**args, **kwargs*)

Provides wavelength calibration information from arc images.

This recipes process a set of arc images obtained in *Arc Calibration* mode and returns the information required to perform wavelength calibration and resampling in other recipes, in the form of a WavelengthCalibration object. The recipe also returns a 2D map of the FWHM of the arc lines used for the calibration, the result images up to dark correction, and the result of the processing up to aperture extraction.

**See also:**

**`megaradrp.products.WavelengthCalibration`** description of WavelengthCalibration product

**`megaradrp.products.LinesCatalog`** description of the catalog of lines

**`megaradrp.processing.aperture`** aperture extraction

**`numina.array.wavecalib.arccalibration`**[20] wavelength calibration algorithm

**`megaradrp.instrument.configs`** instrument configuration

#### Notes

Images provided in *obresult* are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is not None), bias and dark current (if *master_dark* is not None). Images thus corrected are the stacked using the median.

The result of the combination is saved as an intermediate result, named 'reduced_image.fits'. This combined image is also returned in the field *reduced_image* of the recipe result.

The apertures in the 2D image are extracted, using the information in *master_traces*. the result of the extraction is saved as an intermediate result named 'reduced_rss.fits'. This RSS is also returned in the field *reduced_rss* of the recipe result.

For each fiber in the reduced RSS, the peaks are detected and sorted by peak flux. *nlines* is used to select the brightest peaks. If it is a list, then the peaks are divided, by their position, in as many groups as elements in the list and *nlines[0]* peaks are selected in the first group, *nlines[1]* peaks in the second, etc.

The selected peaks are matched against the catalog of lines in *lines_catalog*. The matched lines, the quality of the match and other relevant information is stored in the product WavelengthCalibration object. The wavelength of the matched features is fitted to a polynomial of degree *polynomial_degree*. The coefficients of the polynomial are stored in the final *master_wlcalib* object for each fiber.

**class** **`ArcCalibrationRecipeInput`**(**args, **kwds*)

ArcCalibrationRecipeInput documentation.

**Attributes**

[20] https://numina.readthedocs.io/en/latest/reference/array.html#module-numina.array.wavecalib.arccalibration

**debug_plot** [int, requirement, optional] Save intermediate tracing plots

**extraction_offset** [ListOfType, requirement, optional, default=[0.0]] Offset traces for extraction

**lines_catalog** [MegaraLinesCatalog, requirement] Catalog of lines

**master_apertures** [MultiType, requirement] Apertures information for extraction

**master_bias** [MasterBias, requirement] Master BIAS image

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**master_dark** [MasterDark, requirement, optional] Master DARK image

**nlines** [ListOfType, requirement, optional, default=[20]] Use the 'nlines' brigthest lines of the spectrum

**obresult** [ObservationResultType, requirement] Observation Result

**polynomial_degree** [ListOfType, requirement, optional, default=[5]] Polynomial degree of arc calibration

**store_pdf_with_refined_fits** [int, requirement, optional] Store PDF plot with refined fits for each fiber

**class ArcCalibrationRecipeResult**(**args*, ***kwds*)
ArcCalibrationRecipeResult documentation.

**Attributes**

**fwhm_image** [DataFrameType, product]

**master_wlcalib** [WavelengthCalibration, product]

**qc** [QualityControlProduct, product]

**reduced_image** [ProcessedFrame, product]

**reduced_rss** [ProcessedRSS, product]

**RecipeInput**
alias of `numina.core.metaclass.ArcCalibrationRecipeInput`

**RecipeResult**
alias of `numina.core.metaclass.ArcCalibrationRecipeResult`

**calc_fwhm_of_line**(*row*, *peak_int*, *lwidth=20*)
Compute FWHM of lines in spectra

**model_coeff_vs_fiber**(*data_wlcalib*, *poldeg*, *times_sigma_reject=5*, *debugplot=0*)
Model polynomial coefficients vs. fiber number.

For each polynomial coefficient, a smooth polynomial dependence with fiber number is also computed, rejecting information from fibers which coefficients depart from that smooth variation.

**run**(*rinput*)
Execute the recipe.

**Parameters**

**rinput** [ArcCalibrationRecipe.RecipeInput]

**Returns**

**ArcCalibrationRecipe.RecipeResult**

### 5.1.7 Fiber-flat

**Mode** Fiber-flat

**Usage** Offline, Online

**Key** MegaraFiberFlatImage

**Product** *MasterFiberFlat*

**Recipe** *FiberFlatRecipe*

**Recipe input** `FiberFlatRecipeInput`

**Recipe result** `FiberFlatRecipeResult`

In fiber-fed spectrographs such as MEGARA each optical fiber behaves like a different optical system, and therefore, its optical transmission is different and individual, with different wavelength dependence. In the Preliminary Design phase this mode was named "Lamp fiber flat".

#### Requirements

This observing mode should include the actions to calibrate the low-frequency variations in transmission in between fibers and as a function of wavelength in MEGARA. A fiber-flat should be used to perform this correction and is the result of illuminating the instrument focal plane with a flat source that can be either a continuum (halogen) lamp that is part of the GTC Instrument Calibration Module (ICM) or the twilight sky. The fiber-flat observing mode discussed here assumes that the focal plane is illuminated with a halogen lamp located at the ICM. The ICM beam has to have the same focal ratio arriving to the first MEGARA optical element (the MEGARA telecentricity-correction lens in this case) simulating as much as possible the real GTC mirrors beam at F-C.

These fiber-flat images are also used to trace the fiber spectra on the detector for each specific spectral setup. Finally, they are also useful to verify the status of the optical link between the F-C focal plane and the platform where the spectrographs are located.

This mode requires having the ICM turned on, one of the halogen lamps at the ICM also turned on, to configure the focal-plane cover (at least one of the sides should be open), to have the instrument shutter open, to move the pseudo-slit to that of the instrument mode of choice, to configure the VPH wheel mechanism in order to select the grating to be used, to move the focusing mechanism to the position pre-defined for the specific VPH of choice and to expose a certain time and to readout the detector a series of exposures, being this series the fiber-flat image set.

#### Procedure

The "User" processes an observing block obtained in the observing mode Fiber-flat. This mode includes the required actions to obtain a master fiber-flat field. The master fiber-flat field generated is used in other stages of the data processing.

#### Products

Fiber-flat image sets are to be obtained both as part of the activities related to the verification of the instrument status and for processing data for scientific exploitation.

#### Recipe, inputs and results

**class** `megaradrp.recipes.calibration.flat.`**`FiberFlatRecipe`**(**args*, ***kwargs*)

Process FIBER_FLAT images and create MASTER_FIBER_FLAT product.

This recipe process a set of continuum flat images obtained in **Fiber Flat** mode and returns the master fiber flat product The recipe also returns the result of processing the input images up to slitflat correction. and the result RSS of the processing up to wavelength calibration.

**See also:**

**megaradrp.products.MasterFiberFlat** description of MasterFiberFlat product

**megaradrp.processing.aperture** aperture extraction

*megaradrp.processing.wavecalibration* resampling for wavelength calibration

## Notes

Images provided in *obresult* are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is not None), bias and dark current (if *master_dark* is not None) and corrected from pixel-to-pixel flat if *master_slitflat* is not None. Images thus corrected are the stacked using the median.

The result of the combination is saved as an intermediate result, named 'reduced_image.fits'. This combined image is also returned in the field *reduced_image* of the recipe result.

The apertures in the 2D image are extracted, using the information in *master_traces* and resampled accoding to the wavelength calibration in *master_wlcalib*. The resulting RSS is saved as an intermediate result named 'reduced_rss.fits'. This RSS is also returned in the field *reduced_rss* of the recipe result.

To normalize the *master_fiberflat*, each fiber is divided by a smoothed version (using a Savitzky-Golay filter) of the average of the valid fibers. Finally, all the pixels with information are fiiled with ones. This RSS image is returned in the field *master_fiberflat* of the recipe result.

**class FiberFlatRecipeInput**(**args, **kwds*)
FiberFlatRecipeInput documentation.

**Attributes**

**extraction_offset** [ListOfType, requirement, optional, default=[0.0]] Offset traces for extraction

**master_bias** [MasterBias, requirement] Master BIAS image

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**master_dark** [MasterDark, requirement, optional] Master DARK image

**master_slitflat** [MasterSlitFlat, requirement, optional] Master slit flat calibration

**master_traces** [MultiType, requirement] Apertures information for extraction

**master_wlcalib** [WavelengthCalibration, requirement] Wavelength calibration table

**obresult** [ObservationResultType, requirement] Observation Result

**smoothing_window** [int, requirement, optional, default=31] Window for smoothing (must be odd)

**class FiberFlatRecipeResult**(**args, **kwds*)
FiberFlatRecipeResult documentation.

**Attributes**

**master_fiberflat** [MasterFiberFlat, product]

**qc** [QualityControlProduct, product]

**reduced_image** [ProcessedFrame, product]

**reduced_rss** [ProcessedRSS, product]

**RecipeInput**
alias of `numina.core.metaclass.FiberFlatRecipeInput`

**RecipeResult**
alias of `numina.core.metaclass.FiberFlatRecipeResult`

**run** (*rinput*)
Execute the recipe.

**Parameters**

**rinput** [RecipeInput]

**Returns**

**RecipeResult**

**set_base_headers** (*hdr*)
Set metadata in FITS headers.

### 5.1.8 Twilight fiber-flat

**Mode** Twilight fiber-flat

**Usage** Offline, Online

**Key** MegaraTwilightFlatImage

**Product** *`MasterTwilightFlat`*

**Recipe** *`TwilightFiberFlatRecipe`*

**Recipe input** *`megaradrp.recipes.calibration.twilight.RecipeInput`*

**Recipe result** *`megaradrp.recipes.calibration.twilight.RecipeResult`*

Depending on the final performance of the ICM (provided by the GTC) at F-C the twilight fiber-flat mode (proposed in this section) might be offered as optional to the observer or a must should a proper data reduction be required. In any case this must be always available as an observing mode.

The twilight fiber-flat observing mode should include the actions required to calibrate the low-frequency sensitivity variation in the spatial direction of the detector. In principle, the lamp fiber-flat should suffice to correct the change in sensitivity along both the spatial (fiber-to-fiber relative transmission) and the spectral direction of the system. The latter only combined with flux standard-star observations since the spectral shape of the ICM lamps is not known with enough accuracy.

The twilight fiber-flat is based on the observation of the blank twilight sky. This can safely assume to homogeneously illuminate the entire MEGARA field of view (3.5 arcmin x 3.5 arcmin).

#### Requeriments

The focal plane should be uniformly illuminated with twilight-sky light. As the illumination conditions change during twilight, each image set has a different exposure time. The purpose is to obtain a similar (linear) level of DUs at the detector (counts) under different illumination conditions.

This mode requires having the focal-plane cover configured (at least one of the sides should be open), the instrument shutter open, the telescope tracking, to move the pseudo-slit to that of the instrument mode of choice, to configure the VPH wheel mechanism in order to select the grating to be used, to move the focusing mechanism to the position pre-defined for the specific VPH of choice and to take a series of exposures with different exposure times and to readout the detector for this series of exposures, being these series the twilight image set, each with a different exposure time, but with similar level of counts.

### Procedure

The "User" processes an observing block obtained in the observing mode Twilight Fiber Flat. This mode includes the required actions to obtain a master illumination flat field. The master illumination flat field generated is used in other stages of the data processing.

### Products

Twilight-sky fiber-flat image sets are expected to be obtained as part of the routine calibration activities performed by the observer since are needed for processing any scientific-valid data. Therefore, this observing mode should be considered as part of the "Daily Calibration Modes".

A RSS master illumination flat field, QA flag, a text log file of the processing and a structured text file containing information about the processing.

### Recipe, inputs and results

**class** `megaradrp.recipes.calibration.twilight.`**`RecipeInput`**(**args*, ***kwds*)
RecipeInput documentation.

**Attributes**

**continuum_region** [ListOfType, requirement, optional, default=[1900, 1900]] Subtract this region before normalize the flat-field

**extraction_offset** [ListOfType, requirement, optional, default=[0.0]] Offset traces for extraction

**master_bias** [MasterBias, requirement] Master BIAS image

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**master_dark** [MasterDark, requirement, optional] Master DARK image

**master_fiberflat** [MasterFiberFlat, requirement] Master fiber flat calibration

**master_slitflat** [MasterSlitFlat, requirement, optional] Master slit flat calibration

**master_traces** [MultiType, requirement] Apertures information for extraction

**master_wlcalib** [WavelengthCalibration, requirement] Wavelength calibration table

**normalize_region** [ListOfType, requirement, optional, default=[1900, 2100]] Region used to normalize the flat-field

**obresult** [ObservationResultType, requirement] Observation Result

**class** `megaradrp.recipes.calibration.twilight.`**`RecipeResult`**(**args*, ***kwds*)
RecipeResult documentation.

**Attributes**

**master_twilightflat** [MasterTwilightFlat, product]

**reduced_image** [ProcessedFrame, product]

**reduced_rss** [ProcessedRSS, product]

**class** `megaradrp.recipes.calibration.twilight.`**`TwilightFiberFlatRecipe`**(**args*, ***kwargs*)
Process TWILIGHT_FLAT images and create MASTER_TWILIGHT_FLAT product.

This recipe process a set of continuum flat images obtained in **Twilight Fiber Flat** mode and returns the master twilight flat product The recipe also returns the result of processing the input images up to slitflat correction. and the result RSS of the processing up to wavelength calibration.

**See also:**

`megaradrp.types.MasterTwilightFlat` description of MasterTwilightFlat product

## Notes

Images provided in *obresult* are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is not None), bias and dark current (if *master_dark* is not None) and corrected from pixel-to-pixel flat if *master_slitflat* is not None. Images thus corrected are the stacked using the median.

The result of the combination is saved as an intermediate result, named 'reduced_image.fits'. This combined image is also returned in the field *reduced_image* of the recipe result.

The apertures in the 2D image are extracted, using the information in *master_traces* and resampled according to the wavelength calibration in *master_wlcalib*. Then is divided by the *master_fiberflat*. The resulting RSS is saved as an intermediate result named 'reduced_rss.fits'. This RSS is also returned in the field *reduced_rss* of the recipe result.

To normalize the *master_twilight_flat*, each fiber is divided by the average of the column range given in *normalize_region*. This RSS image is returned in the field *master_twilightflat* of the recipe result.

`class MegaraBaseRecipeInput`(**args*, ***kwds*)
MegaraBaseRecipeInput documentation.

**Attributes**

**obresult** [ObservationResultType, requirement] Observation Result

`class MegaraBaseRecipeResult`(**args*, ***kwds*)
MegaraBaseRecipeResult documentation.

**Attributes**

**qc** [QualityControlProduct, product]

`class RecipeInput`(**args*, ***kwds*)
RecipeInput documentation.

**Attributes**

**continuum_region** [ListOfType, requirement, optional, default=[1900, 1900]] Subtract this region before normalize the flat-field

**extraction_offset** [ListOfType, requirement, optional, default=[0.0]] Offset traces for extraction

**master_bias** [MasterBias, requirement] Master BIAS image

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**master_dark** [MasterDark, requirement, optional] Master DARK image

**master_fiberflat** [MasterFiberFlat, requirement] Master fiber flat calibration

**master_slitflat** [MasterSlitFlat, requirement, optional] Master slit flat calibration

**master_traces** [MultiType, requirement] Apertures information for extraction

**master_wlcalib** [WavelengthCalibration, requirement] Wavelength calibration table

**normalize_region** [ListOfType, requirement, optional, default=[1900, 2100]] Region used to normalize the flat-field

**obresult** [ObservationResultType, requirement] Observation Result

`class RecipeResult`(**args*, ***kwds*)
RecipeResult documentation.

**Attributes**

**master_twilightflat** [MasterTwilightFlat, product]

**reduced_image** [ProcessedFrame, product]

**reduced_rss** [ProcessedRSS, product]

**set_base_headers**(*hdr*)
Set metadata in FITS headers.

### 5.1.9 Bad-pixels mask

**Mode** Bad-pixels mask

**Usage** Offline, Online

**Key** MegaraBadPixelMask

**Product** *MasterBPM*

**Recipe** *BadPixelsMaskRecipe*

**Recipe input** *BadPixelsMaskRecipeInput*

**Recipe result** *BadPixelsMaskRecipeResult*

Although science-grade CCD detectors show very few bad pixels / bad columns there will be a number of pixels (among the ~17 Million pixels in the MEGARA CCD) whose response could not be corrected by means of using calibration images such as dark frames or flat-field images. These pixels, commonly called either dead or hot pixels, should be identified and masked so their expected signal could be derived using dithered images or, alternatively, locally interpolated. While a bad-pixels mask will be generated as part of the AIV activities, an increase in the number of such bad pixels with time is expected. Therefore, we here define an observing mode that the observatory staff could use to generate an updated version of the bad-pixels masks should the number of bad pixels increase significantly.

In the case of fiber-fed spectrographs the fiber flats (either lamp or twilight flats) are not optimal for generating bad-pixels masks as these leave many regions in the CCD not exposed to light. The whole CCD should be illuminated at different intensity levels in order to clearly identify both dead and hot pixels.

#### Requirements

In the case of MEGARA we will offset the pseudo-slit from its optical focus position to ensure that the gaps between fibers are also illuminated when a continuum (halogen) lamp at the ICM is used. The NSC zemax model of the spectrograph indicates that by offsetting 3mm the pseudo-slit we would already obtain a homogenous illumination of the CCD. A series of images with different count levels would be obtained.

This mode requires having the ICM halogen lamp on, the instrument shutter open, to move the pseudo-slit to the open position, to configure the VPH wheel mechanism in order to select the grating to be used, to move the focusing mechanism to the position pre-defined for the specific VPH of choice but offset by 3mm and to expose a certain time and to readout the detector a series of exposures, being this series the slit-flat image set. Note that only one Bad-pixels mask will be used for all spectral setups. The specific choice for the VPH will depend on the actual color of the ICM halogen lamp and on the actual response of the VPHs. In principle, we should choose the VPH at the peak of the lamp spectral energy distribution but we should also consider the fact that the VPH should have the flattest spectral response possible. We call this specific VPH the “BPM VPH”. LR-R and LR-I are currently the best candidates for finally being the BPM VPH.

### Procedure

The "User" processes an observing block obtained in the observing mode Bad-pixels mask. This mode includes the required actions to obtain a bad-pixel mask. The master bad pixel mask generated is used in other stages of the data processing.

### Products

This Bad-pixels mask observing mode will be used only sporadically as it is considered part of the "System Calibration Modes".

A bidimensional mask of bad pixels, a QA flag, a text log file of the processing and a structured text file with information about the processing.

### Recipe, inputs and results

**class** `megaradrp.recipes.calibration.bpm.`**`BadPixelsMaskRecipe`**(**args*, ***kwargs*)

Process defocussed FIBER_FLAT images and create MASTER_BPM product.

This recipe process a set of defocused continuum flat images obtained in **Bad-pixels mask** mode and returns the master bad-pixels mask product.

**See also:**

`numina.array.cosmetics.ccdmask`[21] algorithm to select bad-pixels

*`megaradrp.types.MasterBPM`* description of MasterBPM product

#### Notes

Images provided in *obresult* are trimmed and corrected from overscan, bias and dark current (if *master_dark* is not None). The first half of the images are the stacked using the median and saved as intermediate result 'reduced_image_1.fits'. The second half is also combined and saved as intermediate result 'reduced_image_2.fits'

These two images are passed to the *ccdmask* function, that selects bad-pixels by finding outliers in the ratio of the two images.

The mask is returned in the field *master_bpm* of the recipe result.

**class** **`BadPixelsMaskRecipeInput`**(**args*, ***kwds*)

BadPixelsMaskRecipeInput documentation.

**Attributes**

**master_bias** [MasterBias, requirement] Master BIAS image

**master_dark** [MasterDark, requirement, optional] Master DARK image

**obresult** [ObservationResultType, requirement] Observation Result

**class** **`BadPixelsMaskRecipeResult`**(**args*, ***kwds*)

BadPixelsMaskRecipeResult documentation.

**Attributes**

**master_bpm** [MasterBPM, product]

**qc** [QualityControlProduct, product]

**`RecipeInput`**

alias of `numina.core.metaclass.BadPixelsMaskRecipeInput`

[21] https://numina.readthedocs.io/en/latest/reference/array.html#numina.array.cosmetics.ccdmask

**RecipeResult**
alias of `numina.core.metaclass.BadPixelsMaskRecipeResult`

**set_base_headers**(*hdr*)
Set metadata in FITS headers.

**validate_input**(*recipe_input*)
Validate input of the recipe.

The number of frames in *recipe_input.obresult* must be even.

**Raises**

**numina.exceptions.ValidationError** If the number of frames in *obresult* is odd

**class** BadPixelsMaskRecipe.**BadPixelsMaskRecipeInput**(**args*, ***kwds*)
BadPixelsMaskRecipeInput documentation.

**Attributes**

**master_bias** [MasterBias, requirement] Master BIAS image

**master_dark** [MasterDark, requirement, optional] Master DARK image

**obresult** [ObservationResultType, requirement] Observation Result

**class** BadPixelsMaskRecipe.**BadPixelsMaskRecipeResult**(**args*, ***kwds*)
BadPixelsMaskRecipeResult documentation.

**Attributes**

**master_bpm** [MasterBPM, product]

**qc** [QualityControlProduct, product]

### 5.1.10 Linearity test

**Mode** Linearity test

**Usage** Offline, Online

**Key** MegaraLinearityTest

**Product**

**Recipe** `LinearityTestRecipe`

**Recipe input** `LinearityTestRecipeInput`

**Recipe result** `LinearityTestRecipeResult`

Although the linearity of the MEGARA CCD are well characterized at the LICA lab already, it might be advisable to generate linearity test frames both as part of the AIV activities and after changes in the MEGARA DAS.

The MEGARA e2V 231-84 CCD offers a full-well capacity of 350,000 ke-. Linearity tests carried out in instruments already using this type of CCD indicate a linearity better than ±0.4% at 100 kpix/sec in the range between 140 to 40,000 e- (Reiss et al. 2009 for MUSE@VLT). Given these good linearity results (up to 40,000 e-) and considering the fact that at the spectral resolutions of MEGARA we will rarely reach those signals from astronomical targets linearity can be considered negligible. Despite these facts, it is advisable to carry out this kind of tests both at the lab and at the telescope on the MEGARA CCD itself.

While Linearity tests will be generated as part of the characterization activities at the lab, the use of the ICM would also allow carrying them out as part of AIV activities and routinely as part of the "System Calibration Modes". Therefore, we define here an observing mode that the observatory staff could use to generate updated Linearity tests should these be needed.

In the case of fiber-fed spectrographs the fiber flats (either lamp or twilight flats) are not optimal for carrying out Linearity tests as these leave many regions in the CCD not exposed to light. The whole CCD should be illuminated at different intensity levels in order for properly carrying out these tests.

#### Requirements

In the case of MEGARA we will offset the pseudo-slit from its optical focus position to ensure that the gaps between fibers are also illuminated when a continuum (halogen) lamp at the ICM is used. The NSC zemax model of the spectrograph indicates that by offsetting 3mm the pseudo-slit we would already obtain a homogenous illumination of the CCD. A series of images with different count levels would be obtained.

This mode requires having the ICM halogen lamp on, the instrument shutter open, to move the pseudo-slit to the open position, to configure the VPH wheel mechanism in order to select the grating to be used, to move the focusing mechanism to the position pre-defined for the specific VPH of choice and to expose a certain time and to readout the detector a series of exposures, being this series the slit-flat image set. Note that the Linearity test will be done using only one spectral setup as this is independent of the VPH of use. The specific choice for the VPH will depend on the actual color of the ICM halogen lamp and on the actual response of the VPHs. In principle, we should choose the VPH at the peak of the lamp spectral energy distribution but we

#### Procedure

#### Products

This Linearity-test observing mode will be used only sporadically as it is considered part of the "System Calibration Modes".

#### Recipe, inputs and results

### 5.1.11 Standard star with the LCB IFU

**Mode** Standard star with the LCB IFU

**Usage** Offline, Online

**Key** MegaraLcbStdStar

**Recipe** *`LCBStandardRecipe`*

**Recipe input** `LCBStandardRecipeInput`

**Recipe result** `LCBStandardRecipeResult`

This observing mode includes the required actions to obtain those calibration images needed to correct for the variation in the response of the system along the spectral direction. This signature is manifested by a change in the conversion factor between the energy surface density hitting the telescope primary mirror and the DUs per CCD pixel with wavelength. Its effect is already present in the original data but it could get modified during the reduction process, e.g. after the fiber-flat correction is applied.

The flux calibration is performed by observing one or several spectrophotometric stars with the same instrument configuration that for the scientific observations. Depending on the number of standard stars observed and on the weather conditions (mainly transparency) two different types of calibration could be achieved:

- Absolute-flux calibration: The weather conditions during the night should be photometric and a number of spectrophotometric standard stars at different airmasses should be observed. This allows to fully correct from DUs per CCD pixel to energy surface density (typically in erg s-1 cm-2 Å-1) incident at the top of the atmosphere. If only one single standard star is observed (at the airmass of the science object) this correction allows deriving the energy surface density hitting the telescope primary mirror exclusively, unless an atmospheric extinction curve for the observatory and that particular night is assumed. In order to properly flux-calibrate scientific observations at all airmasses several stars should be observed during the night.

- Relative-flux calibration: If the weather conditions are not photometric this correction only allows normalizing the DUs per CCD pixel along the spectral direction so the conversion to incident energy at the top of the atmosphere is the same at all wavelengths. In order for this calibration to be valid the assumption that the effect of the atmosphere (including atmospheric cirrus and possibly thick clouds) on the wavelength dependence of this correction is that given by the atmospheric extinction curve adopted.

Since the observing sequence needed for both types of flux calibration is identical only one observing mode (standard star) needs to be defined.

We will use this same observing mode also for the observation of either telluric standards or radial-velocity standards. The former are needed to correct for the presence of telluric absorptions mainly in the red part of the spectrum and are achieved by means of observing A-type stars at the same airmass and very close in time to the corresponding scientific observation. The latter can used to determine a precise zero point velocity for the instrument at a specific night and to verify its stability from night to night and season to season.

### Requirements

This mode requires the entire flux of the spectrophotometric standard star to be recovered (even if the star is a telluric or radial-velocity standard), especially when an absolute-flux calibration is needed, so the LCB IFU bundle must be used. The FOV of the LCB IFU is large enough for these observations to be carried out with one of the sides of the focal-plane cover closed. When this calibration is aimed for a set of Fiber-MOS scientific observations, complementary observations of standard stars through the Fiber-MOS minibundles might be also required. This allows verifying the quality and stability of the calibration when two different pseudo-slits are used. Such observing mode is described later.

This mode requires having the focal-plane cover configured (at least one of the sides should be open), the instrument shutter open, to configure the VPH mechanism to select the grating to be used, to set the instrument mode to LCB IFU, to move the focusing mechanism to the position pre-defined for the specific VPH of choice, and to expose a certain time and to readout the detector in a series of exposures, being this series the image set containing the spectral energy distribution of the spectrophotometric standard star.

In order to distribute the flux from the star across multiple spaxels in the LCB IFU bundle (particularly important in the case of very bright spectrophotometric standard stars) we might also need to apply a small drift motion (typically of a few arcsec per second) to one of the telescope axes at the start of the observation or, more likely, slightly defocus the telescope.

### Products

Standard star image sets are to be obtained only as part of the routine calibration activities performed by the observer and that are needed for processing data for scientific exploitation.

### Recipe, inputs and results

**class** `megaradrp.recipes.calibration.lcbstdstar.`**`LCBStandardRecipe`**(**args, **kwargs*)

Process LCB Standard Star Recipe.

This recipe processes a set of images obtained in **LCB Stardard Star image** mode and returns the total flux of the star.

**See also:**

*`megaradrp.recipes.calibration.mosstdstar.MOSStandardRecipe`*

### Notes

Images provided by *obresult* are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is not None), bias, dark current (if *master_dark* is not None) and slit-flat (if *master_slitflat* is not None).

Images thus corrected are then stacked using the median. The result of the combination is saved as an intermediate result, named 'reduced_image.fits'. This combined image is also returned in the field *reduced_image* of the recipe result.

The apertures in the 2D image are extracted, using the information in *master_traces* and resampled according to the wavelength calibration in *master_wlcalib*. Then is divided by the *master_fiberflat*. The resulting RSS is saved as an intermediate result named 'reduced_rss.fits'. This RSS is also returned in the field *reduced_rss* of the recipe result.

The sky is subtracted by combining the the fibers marked as *SKY* in the fibers configuration. The RSS with sky subtracted is returned ini the field *final_rss* of the recipe result.

The flux of the star is computed by adding summing the fibers in *nrings* around the central spaxel containing the star and returned as *star_spectrum*.

**class LCBStandardRecipeInput**(**args*, ***kwds*)
LCBStandardRecipeInput documentation.

**Attributes**

**extraction_offset** [ListOfType, requirement, optional, default=[0.0]] Offset traces for extraction

**ignored_sky_bundles** [ListOfType, requirement, optional] Ignore these sky bundles

**master_bias** [MasterBias, requirement] Master BIAS image

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**master_dark** [MasterDark, requirement, optional] Master DARK image

**master_fiberflat** [MasterFiberFlat, requirement] Master fiber flat calibration

**master_sensitivity** [MasterSensitivity, requirement, optional] Master sensitivity for flux calibration

**master_slitflat** [MasterSlitFlat, requirement, optional] Master slit flat calibration

**master_traces** [MultiType, requirement] Apertures information for extraction

**master_twilight** [MasterTwilightFlat, requirement, optional] Master twlight flat calibration

**master_wlcalib** [WavelengthCalibration, requirement] Wavelength calibration table

**nrings** [int, requirement, optional, default=3] Number of rings to extract the star

**obresult** [ObservationResultType, requirement] Observation Result

**position** [list, requirement] Position of the reference object

**reference_extinction** [ReferenceExtinctionTable, requirement] Reference extinction

**reference_spectrum** [ReferenceSpectrumTable, requirement] Spectrum of reference star

**reference_spectrum_velocity** [float, requirement, optional] Radial velocity (km/s) of reference spectrum

**relative_threshold** [float, requirement, optional, default=0.3] Threshold for peak detection

**sigma_resolution** [float, requirement, optional, default=20.0] sigma Gaussian filter to degrade resolution

**class LCBStandardRecipeResult** (**args, **kwds*)
LCBStandardRecipeResult documentation.

**Attributes**

**fiber_ids** [ArrayType, product]

**final_rss** [ProcessedRSS, product]

**master_sensitivity** [MasterSensitivity, product]

**qc** [QualityControlProduct, product]

**reduced_image** [ProcessedFrame, product]

**reduced_rss** [ProcessedRSS, product]

**sky_rss** [ProcessedRSS, product]

**star_spectrum** [ProcessedSpectrum, product]

**RecipeInput**
alias of `numina.core.metaclass.LCBStandardRecipeInput`

**RecipeResult**
alias of `numina.core.metaclass.LCBStandardRecipeResult`

### 5.1.12 Standard star with the Fiber MOS

**Mode** Standard star with the FIBER MOS

**Usage** Offline, Online

**Key** MegaraMosStdStar

**Recipe** *`MOSStandardRecipe`*

**Recipe input** `MOSStandardRecipeInput`

**Recipe result** `MOSStandardRecipeResult`

This observing mode includes the required actions to obtain those calibration images needed to correct for the variation in the response of the system along the spectral direction. The difference between this mode and the two precedent observing modes is that in this case the spectrophotometric standard star is observed through one of the robotic positioners of the Fiber-MOS subsystem.

As in Standard star with the IFUs observing modes, the calibration is performed by observing one or several spectrophotometric stars with the same instrument configuration that for the scientific observations. Depending on the number of standard stars observed and the weather conditions two different types of calibration could be achieved, absolute or relative. In the case of the former calibration an aperture correction should be applied to take into account the possible flux losses from the standard stars when observed through one of the ~1.6-arcsec-wide robotic positioners.

#### Requirements

This mode requires having the focal-plane cover configured, the instrument shutter open, to configure the VPH mechanism to select the grating to be used, to set the instrument mode to Fiber MOS, to move the focusing mechanism to the position pre-defined for the specific VPH of choice, to move one of the robotic positioners to the position of the spectrophotometric standard star (other positioners could be also moved if needed) and to expose a certain time and to readout the detector in a series of exposures, being this series the image set containing the spectral energy distribution of the spectrophotometric standard star.

This observing mode could still be carried out with one of the sides of the focal-plane cover closed. However, as the (commonly rather bright) spectrophotometric standard star is the only object of interest in the field, the other positioners would not be observing scientific targets, so the level of cross-talk between these and the positioner devoted to the standard star should be negligible. Thus, the use of the focal-plane cover, although considered, is not recommended for this specific observing mode.

In order to place the robotic positioner(s) on the corresponding target(s) a set of input catalogues previously generated by the observer using MOPSS (MEGARA Observing Preparation Software Suite) are needed.

### Products

Standard star image sets are to be obtained only as part of the routine calibration activities performed by the observer that are needed for processing data for scientific exploitation.

### Recipe, inputs and results

**class** megaradrp.recipes.calibration.mosstdstar.**MOSStandardRecipe**(**args*, ***kwargs*)

Process MOS Standard Star Recipe.

This recipe processes a set of images obtained in **MOS Stardard Star image** mode and returns the total flux of the star.

**See also:**

*megaradrp.recipes.calibration.lcbstdstar.LCBStandardRecipe*

#### Notes

Images provided by *obresult* are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is not None), bias, dark current (if *master_dark* is not None) and slit-flat (if *master_slitflat* is not None).

Images thus corrected are then stacked using the median. The result of the combination is saved as an intermediate result, named 'reduced_image.fits'. This combined image is also returned in the field *reduced_image* of the recipe result.

The apertures in the 2D image are extracted, using the information in *master_traces* and resampled according to the wavelength calibration in *master_wlcalib*. Then is divided by the *master_fiberflat*. The resulting RSS is saved as an intermediate result named 'reduced_rss.fits'. This RSS is also returned in the field *reduced_rss* of the recipe result.

The sky is subtracted by combining the the fibers marked as *SKY* in the fibers configuration. The RSS with sky subtracted is returned ini the field *final_rss* of the recipe result.

The flux of the star is computed by adding the 7 fibers corresponding to the bundle containing the star and returned as *star_spectrum*.

**class MOSStandardRecipeInput**(**args*, ***kwds*)

MOSStandardRecipeInput documentation.

**Attributes**

**extraction_offset** [ListOfType, requirement, optional, default=[0.0]] Offset traces for extraction

**ignored_sky_bundles** [ListOfType, requirement, optional] Ignore these sky bundles

**master_bias** [MasterBias, requirement] Master BIAS image

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**master_dark** [MasterDark, requirement, optional] Master DARK image

**master_fiberflat** [MasterFiberFlat, requirement] Master fiber flat calibration

**master_sensitivity** [MasterSensitivity, requirement, optional] Master sensitivity for flux calibration

**master_slitflat** [MasterSlitFlat, requirement, optional] Master slit flat calibration

**master_traces** [MultiType, requirement] Apertures information for extraction

**master_twilight** [MasterTwilightFlat, requirement, optional] Master twlight flat calibration

**master_wlcalib** [WavelengthCalibration, requirement] Wavelength calibration table

**obresult** [ObservationResultType, requirement] Observation Result

**position** [list, requirement] Position of the reference object

**reference_extinction** [ReferenceExtinctionTable, requirement] Reference extinction

**reference_spectrum** [ReferenceSpectrumTable, requirement] Spectrum of reference star

**reference_spectrum_velocity** [float, requirement, optional] Radial velocity of reference spectrum

**relative_threshold** [float, requirement, optional, default=0.3] Threshold for peak detection

**sigma_resolution** [float, requirement, optional, default=20.0] sigma Gaussian filter to degrade resolution

**class MOSStandardRecipeResult**(**args*, ***kwds*)

MOSStandardRecipeResult documentation.

**Attributes**

**fiber_ids** [ArrayType, product]

**final_rss** [ProcessedRSS, product]

**master_sensitivity** [MasterSensitivity, product]

**qc** [QualityControlProduct, product]

**reduced_image** [ProcessedFrame, product]

**reduced_rss** [ProcessedRSS, product]

**sky_rss** [ProcessedRSS, product]

**star_spectrum** [ProcessedSpectrum, product]

**RecipeInput**

alias of `numina.core.metaclass.MOSStandardRecipeInput`

**RecipeResult**

alias of `numina.core.metaclass.MOSStandardRecipeResult`

## 5.2 Auxiliary Modes

The auxiliary modes refer to those tasks that are carried out in preparation for other observing modes and that are needed to ensure the quality of the MEGARA observations. In general, these observing modes are not useful per se, neither for straight scientific exploitation nor even data-processing purposes.

These Auxiliary modes are intended to (1) allow the observatory staff to prepare the instrument for its optimal exploitation or (2) to improve the performance of the telescope procedures in terms of the target acquisition and focus.

With regard to the former, the following modes are defined:

- Telescope focus
- Spectrograph focus

These observing modes might have to be run before any observing run (in case of visitor mode observations) and certainly after a long period of inactivity of the instrument. Below we provide a detailed description of each of these observing modes. Regarding the auxiliary observing modes defined to improve the default telescope procedures these are:

- (The two previous observing modes)
- Fine acquisition with the LCB IFU
- Fine acquisition with the Fiber MOS

These latter three observing modes should be run before any observing run (in case of visitor mode observations) and after a long period of inactivity of the instrument; at least in the instrument mode to be used (LCB or MOS).

### 5.2.1 Telescope focus

**Mode** Telescope Focus

**Usage** Online

**Key** MegaraFocusTelescope

**Recipe** *`FocusTelescopeRecipe`*

**Recipe input** *`FocusTelescopeRecipeInput`*

**Recipe result** *`FocusTelescopeRecipeResult`*

This observing mode includes the required actions to focus GTC using MEGARA. A bright point source should be identified for this purpose. This mode is an alternative to obtain the best focus of the telescope using the A&G system. Since this mode requires having precise information on the spatial distribution of the flux coming from a point source in a region a few arcsec in diameter the use of the LCB IFU bundle.

#### Requirements

This mode requires the observation of a bright point source on the sky continuously while the observing mode is being run. Although photometric conditions are not needed the transparency should allow to properly measure the source FWHM on the sky in any exposure of the series. The FOV of the LCB IFU is large enough for these observations to be carried out with one of the sides of the focal-plane cover closed. However, as the default configuration of the instrument with the focal-plane cover in the open position the alignment will be more likely carried out with the cover in that position.

This mode requires having the focal-plane cover configured (at least one of the sides should be open; the most likely configuration will be with the focal-plane cover fully open), the instrument shutter open, to configure the VPH mechanism to select the grating to be used, to set the instrument mode to LCB IFU, to move the focusing mechanism to the position pre-defined for the specific VPH of choice, to expose a certain time and to readout the detector in a series of exposures, being this series the telescope focus image set. A pause in between every exposure in the series should be introduced in order to give time for the M2 to adjust each new focus position in the series and for the M2 control system to inform about its new position, which should then re-start the observing sequence.

### Products

The observatory staff should obtain the telescope focus image sets as part of standard preparatory observations (according to GRANTECAN this is usually done every night). Should the focus offset between the ASG and SFS arms (or an imaging instrument in other focus) and that of the MEGARA FC be stable overtime, the use of this mode would be limited to the early stages of characterization of the optimal telescope configuration for MEGARA, or after major changes in the instrument, or if any problem arises. The observatory staff should obtain the telescope focus image sets as part of standard preparatory observations obtained at twilight (although not necessarily every night) or when problems with the telescope focus model are suspected. Also, telescope focus should be revised periodically (e.g. monthly) to correct for potential temperature effects. In general, Auxiliary modes will be typically run once every observing run (e.g. the fine-acquisition ones) or, in the best (most relaxed) case, after a long period of inactivity.

### Recipe, inputs and results

`class megaradrp.recipes.auxiliary.focustel.`**`FocusTelescopeRecipe`**`(`**args, **kwargs*`)`

Process telescope focus images and find best focus.

This recipe process a set of focus images obtained in **Focus Telescope** mode and returns an estimation of the telescope best focus.

**See also:**

*`megaradrp.recipes.auxiliary.focusspec.FocusSpectrographRecipe`* recipe to measure the focus of the spectrograph

#### Notes

Images provided in *obresult* are grouped by the value of their FOCUST keyword. Groups of images are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is not None), bias and dark current (if *master_dark* is not None). Each group is then stacked using the median.

The result of the combination is saved as an intermediate result, named 'focus2d-#focus.fits', with #focus being the value of the focus of each group. Apertures are extracted in each combined image, and the resulting RSS file is saved as an intermediate result, named 'focus1-#focus.fits'.

For each image, the FWHM of the object at *position* is computed.

Then, the FWHM is fitted to a 2nd degree polynomial, and the focus corresponding to its minimum is obtained and returned in *focus_table*

`class` **`FocusTelescopeRecipeInput`**`(`**args, **kwds*`)`

FocusTelescopeRecipeInput documentation.

**Attributes**

**extraction_offset** [ListOfType, requirement, optional, default=[0.0]] Offset traces for extraction

**ignored_sky_bundles** [ListOfType, requirement, optional] Ignore these sky bundles

**master_bias** [MasterBias, requirement] Master BIAS image

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**master_dark** [MasterDark, requirement, optional] Master DARK image

**master_fiberflat** [MasterFiberFlat, requirement] Master fiber flat calibration

**master_sensitivity** [MasterSensitivity, requirement, optional] Master sensitivity for flux calibration

**master_slitflat** [MasterSlitFlat, requirement, optional] Master slit flat calibration

**master_traces** [MultiType, requirement] Apertures information for extraction

**master_twilight** [MasterTwilightFlat, requirement, optional] Master twlight flat calibration

**master_wlcalib** [WavelengthCalibration, requirement] Wavelength calibration table

**obresult** [ObservationResultType, requirement] Observation Result

**position** [list, requirement] Position of the reference object

**reference_extinction** [ReferenceExtinctionTable, requirement, optional] Reference extinction

**relative_threshold** [float, requirement, optional, default=0.3] Threshold for peak detection

**class FocusTelescopeRecipeResult** (**args, **kwds*)
FocusTelescopeRecipeResult documentation.

**Attributes**

**focus_table** [float, product]

**qc** [QualityControlProduct, product]

**RecipeInput**
alias of `numina.core.metaclass.FocusTelescopeRecipeInput`

**RecipeResult**
alias of `numina.core.metaclass.FocusTelescopeRecipeResult`

**reorder_and_fit** (*all_images*)
Fit all the values of FWHM to a 2nd degree polynomial and return minimum.

**run_on_image** (*img, coors*)
Extract spectra, find peaks and compute FWHM.

### 5.2.2 Spectrograph focus

**Mode** Spectrograph Focus

**Usage** Online

**Key** MegaraFocusSpectrograph

**Recipe** *FocusSpectrographRecipe*

**Recipe input** *FocusSpectrographRecipeInput*

**Recipe result** *FocusSpectrographRecipeResult*

This mode sequence includes the required actions to focus the MEGARA spectrograph. The arc lamps from the ICM will be used for this purpose. MEGARA includes a focusing mechanism at the position of the pseudo-slits. The best image quality for the spectrograph is achieved by using a different focus position for each disperser element (VPH). The focus position is independent of the instrument mode in use (TBC; see next paragraph).

As said above, the two pseudo-slits (LCB and MOS) have to be at the same exact place at the spectrograph entrance to yield the same focus position. If this is not the case MEGARA can correct from this effect by focusing with a fixed offset for all the VPHs. This value has to be added to the nominal focusing position of a VPH with the pseudo-slit used as reference and shall be taken into account in the MEGARA Control System look-up table. Whether these should be one look-up table (i.e. a different focus compromise) for each focal-plane cover configuration is TBD.

### Requirements

This mode requires having the focal-plane cover configured (at least one of the sides should be open), the instrument shutter open, to configure the VPH mechanism to select the grating to be used, to set the instrument mode to use, to move the focusing mechanism to the pre-defined focus position for specific VPH of choice, to expose a certain time and to readout the detector in a series of exposures, being this series the spectrograph focus image-sets. The focusing mechanism should be able to change its position in between every two exposures in the series. A pause in between every exposure should be introduced in order to give time for the focusing mechanism to adjust to the new focus position in the series and for the MEGARA control system to inform about its new position, which should then re-start the observing sequence (see below). Whether the focus positions (or range) for each VPH are already pre-defined is TBD.

As part of the on-line quick-look software all images in the series should be pre-processed and several spectra along the pseudo-slit should be extracted and analyzed. This analysis should include the computation of the FWHM of a few unresolved spectra lines at different wavelengths. This software should then decide based on the FWHM values computed at different wavelengths and positions along the pseudo-slit the best focus compromise. The best focus obtained for the VPH of choice should then be stored and used to determine the best foci for all spectral configurations (and instrument modes; TBC).

### Procedure

Spectrograph focus image sets through; at least, one of the MEGARA VPHs should be obtained at the beginning of every observing night by either the observer or the staff of the observatory (TBD). Once a VPH is checked, the rest of the values could be corrected relative to this one. It is expected that minor focus corrections should be done as the temperature changes. This could be modeled in further phases and checked at laboratory and/or at the telescope. The observatory staff should obtain an entire sequence of spectrograph focus image sets through all VPHs (and instrument modes; TBC) after major changes in the instrument, long periods of inactivity or when the relative-focus prescriptions (i.e. the spectrograph focus model) are suspected to be inaccurate.

The focus difference (obtained by measuring a particular VPH) will provide the offset focus (due to temperature) and this value will be the same for all VPHs. The Control System will be prepared to update the look-up table with this offset focus value due to temperature.

### Products

The best focus, the goodness of the fit of the best focus, a table with the FWHM of the spectral line corresponding to each focus, position along the slit and wavelength, the collapsed PSFs, QA flag, a text log file of the processing and a structured text file containing information about the processing.

### Recipe, inputs and results

**class** `megaradrp.recipes.auxiliary.focusspec.`**`FocusSpectrographRecipe`**(**args, **kwargs*)

Process spectrograph focus images and find best focus.

This recipe process a set of focus images obtained in **Focus Spectrograph** mode and returns different measurements of the spectrograph focus along de detector.

**See also:**

*megaradrp.recipes.auxiliary.focustel.FocusTelescopeRecipe* recipe to measure the focus of the telescope

### Notes

Images provided in *obresult* are grouped by the value of their FOCUS keyword. Groups of images are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is not None), bias and dark current (if *master_dark* is not None). Each group is then stacked using the median.

The result of the combination is saved as an intermediate result, named 'focus2d-#focus.fits', with #focus being the value of the focus of each group. Apertures are extracted in each combined image, and the resulting RSS file is saved as an intermediate result, named 'focus1-#focus.fits'.

For each image, peaks are detected every *nfibers* fibers, and their position, peak flux and FWHM is computed. The image with median focus is taken as reference image, and the peaks of every other image are matched against it.

Then, for each line matched in the series of images, its FWHM is fitted to a 2nd degree polynomial, and the focus corresponding to its minimum is obtained.

The recipe returns:

- ***focus_table*: a table with (x,y,best_focus) for each matched peak,** with x,y measured in the reference bidimensional image
- ***focus_wavelength*: a structure containing measurements of every** matched peak in each image
- ***focus_image*: a bidimensional image representing the spatial** variation of the best focus, using a Voronoi diagram.

**class FocusSpectrographRecipeInput**(**args*, ***kwds*)
FocusSpectrographRecipeInput documentation.

**Attributes**

**extraction_offset** [ListOfType, requirement, optional, default=[0.0]] Offset traces for extraction

**master_bias** [MasterBias, requirement] Master BIAS image

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**master_dark** [MasterDark, requirement, optional] Master DARK image

**master_traces** [MultiType, requirement] Apertures information for extraction

**master_wlcalib** [WavelengthCalibration, requirement] Wavelength calibration table

**nfibers** [int, requirement, optional, default=10] The results are sampled every nfibers

**obresult** [ObservationResultType, requirement] Observation Result

**tsigma** [int, requirement, optional, default=50] Scale factor for row threshold

**class FocusSpectrographRecipeResult**(**args*, ***kwds*)
FocusSpectrographRecipeResult documentation.

**Attributes**

**focus_image** [ProcessedFrame, product]

**focus_table** [ArrayType, product]

**focus_wavelength** [FocusWavelength, product]

**qc** [QualityControlProduct, product]

### 5.2.3 Fine acquisition with the LCB IFU

**Mode** LCB Acquisition

**Usage** Online

**Key** MegaraLcbAcquisition

**Recipe** *AcquireLCBRecipe*

**Recipe input** *RecipeInput*

**Recipe result** *RecipeResult*

This mode sequence includes the required actions to acquire a target with known celestial coordinates and place it at a reference position inside the LCB IFU instrument mode. The reference position for each mode is defined as the center of the fibers (or its associated microlens) that is closest to the bundle footprint geometrical center. In the case of the LCB the reference position will depend on the focal-plane cover configuration. This mode is a refinement of acquisition performed by the telescope or A&G systems.

#### Requirements

This mode requires having the focal-plane cover configured, the instrument shutter open, to configure the VPH mechanism to select the grating to be used, to set the instrument mode to LCB, to move the focusing mechanism to the position pre-defined for the specific VPH of choice, and to expose a certain time and to readout the detector in a series of exposures, being this series the fine acquisition image set.

As part of the MEGARA on-line quick-look software the image (or images) obtained as part of this observing mode should be processed and the spectra extracted so to determine the position of the centroid of the target in the corresponding field of view. A view of the field should be also produced in order to evaluate whether or not the angle of the Folded-Cass rotator matches that specified by the observer.

#### Products

Fine acquisition image sets should be obtained at the beginning of the observing night by either the observer or the staff of the observatory (TBD) or every time a problem with the telescope absolute pointing is suspected. Such image sets should be also obtained when an absolute positioning precision of the order of a fraction of the spaxel size is required, better than 0.62 arcsec in this case for the LCB.

The observatory staff should decide whether or not the corrections derived must be applied to the acquisition of other targets during the same observing night or exclusively to the target currently being observed.

#### Recipe, inputs and results

**class** megaradrp.recipes.auxiliary.acquisitionlcb.**AcquireLCBRecipe**(**args*, ***kwargs*)

Process Acquisition LCB images.

This recipe processes a set of acquisition images obtained in **LCB Acquisition** mode and returns the offset and rotation required to center the fiducial object in its reference positions.

**See also:**

*megaradrp.recipes.auxiliary.acquisitionmos.AcquireMOSRecipe*

### Notes

Images provided by *obresult* are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is not None), bias, dark current (if *master_dark* is not None) and slit-flat (if *master_slitflat* is not None).

Images thus corrected are the stacked using the median. The result of the combination is saved as an intermediate result, named 'reduced_image.fits'. This combined image is also returned in the field *reduced_image* of the recipe result.

The apertures in the 2D image are extracted, using the information in *master_traces* and resampled according to the wavelength calibration in *master_wlcalib*. Then is divided by the *master_fiberflat*. The resulting RSS is saved as an intermediate result named 'reduced_rss.fits'. This RSS is also returned in the field *reduced_rss* of the recipe result.

The sky is subtracted by combining the the fibers marked as *SKY* in the fibers configuration. The RSS with sky subtracted is returned ini the field *final_rss* of the recipe result.

Then, the centroid of the fiducial object nearest to the center of the field is computed. The offset needed to center the fiducial object in the center of the LCB is returned.

`class AcquireLCBRecipeInput`(**args*, ***kwds*)
AcquireLCBRecipeInput documentation.

**Attributes**

**extraction_offset** [ListOfType, requirement, optional, default=[0.0]] Offset traces for extraction

**extraction_region** [ListOfType, requirement, optional, default=[1000, 3000]] Region used to compute a mean flux

**ignored_sky_bundles** [ListOfType, requirement, optional] Ignore these sky bundles

**master_bias** [MasterBias, requirement] Master BIAS image

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**master_dark** [MasterDark, requirement, optional] Master DARK image

**master_fiberflat** [MasterFiberFlat, requirement] Master fiber flat calibration

**master_sensitivity** [MasterSensitivity, requirement, optional] Master sensitivity for flux calibration

**master_slitflat** [MasterSlitFlat, requirement, optional] Master slit flat calibration

**master_traces** [MultiType, requirement] Apertures information for extraction

**master_twilight** [MasterTwilightFlat, requirement, optional] Master twlight flat calibration

**master_wlcalib** [WavelengthCalibration, requirement] Wavelength calibration table

**obresult** [ObservationResultType, requirement] Observation Result

**points** [ListOfType, requirement, optional, default=[(0, 0)]] Coordinates

**reference_extinction** [ReferenceExtinctionTable, requirement, optional] Reference extinction

**relative_threshold** [float, requirement, optional, default=0.3] Threshold for peak detection

`class AcquireLCBRecipeResult`(**args*, ***kwds*)
AcquireLCBRecipeResult documentation.

**Attributes**

**final_rss** [ProcessedRSS, product]

**offset** [list, product]

**qc** [QualityControlProduct, product]

**reduced_image** [ProcessedFrame, product]

**reduced_rss** [ProcessedRSS, product]

**rotang** [float, product]

**RecipeInput**
alias of `numina.core.metaclass.AcquireLCBRecipeInput`

**RecipeResult**
alias of `numina.core.metaclass.AcquireLCBRecipeResult`

### 5.2.4 Fine acquisition with the Fiber MOS

**Mode** MOS Acquisition

**Usage** Online

**Key** MegaraLcbAcquisition

**Recipe** *AcquireMOSRecipe*

**Recipe input** *RecipeInput*

**Recipe result** *RecipeResult*

The sequence for this observing mode includes the required actions to acquire a list of targets with known celestial coordinates and place each target at the center of a different robotic positioner. The information on the assignment of targets and positioners is included in the form of a set of input catalogues generated off-line by the MEGARA Observing Preparation Software Suite (MOPSS). The reference position for each positioner is the center of the central fiber of the 7-fiber minibundle. This mode is a refinement of the acquisition performed by the telescope or A&G systems.

#### Requirements

This mode requires having the focal-plane cover configured, the instrument shutter open, to configure the VPH mechanism to select the grating to be used, to set the instrument mode to Fiber MOS, to move the focusing mechanism to the position pre-defined for the specific VPH of choice, to move all robotic positioners with a target associated in the input catalogues to the position of the corresponding target and to expose a certain time and to readout the detector in a series of exposures, being this series the Fiber-MOS fine acquisition image set.

As part of the MEGARA on-line quick-look software, the image (or images) obtained should be processed and the spectra extracted so to determine the position of the centroid of a number of reference targets included in the corresponding field of view and identified as such in the set of input catalogues used for this observing mode. A minimum of three reference sources should be included in each Fiber MOS configuration block in order for this observing mode to generate a solution. The quick-look software should compare the expected and the actual positions of these reference sources in order to determine the best-fitting set of offsets (both in X and Y) and rotation angle to apply to the telescope and Folded-Cass rotator, respectively, to then continue with one of the scientific observing modes described in next Section.

#### Products

Fine acquisition image sets should be obtained by the observer at the beginning of the observation of each field with the Fiber MOS. The observatory staff should decide whether or not the corrections derived (telescope offset and Folded-Cass rotator angle) must be applied to the acquisition of other fields with the Fiber MOS during the same observing night or exclusively to the target currently being observed.

## Recipe, inputs and results

**class** megaradrp.recipes.auxiliary.acquisitionmos.**AcquireMOSRecipe**(**args*, ***kwargs*)

Process Acquisition MOS images.

This recipe processes a set of acquisition images obtained in **MOS Acquisition** mode and returns the offset and rotation required to center the fiducial objects in their reference positions.

**See also:**

*megaradrp.recipes.auxiliary.acquisitionlcb.AcquireLCBRecipe*

**numina.array.offrot**[22] Kabsch algorithm for offset and rotation

### Notes

Images provided by *obresult* are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is not None), bias, dark current (if *master_dark* is not None) and slit-flat (if *master_slitflat* is not None).

Images thus corrected are the stacked using the median. The result of the combination is saved as an intermediate result, named 'reduced_image.fits'. This combined image is also returned in the field *reduced_image* of the recipe result.

The apertures in the 2D image are extracted, using the information in *master_traces* and resampled according to the wavelength calibration in *master_wlcalib*. Then is divided by the *master_fiberflat*. The resulting RSS is saved as an intermediate result named 'reduced_rss.fits'. This RSS is also returned in the field *reduced_rss* of the recipe result.

The sky is subtracted by combining the the fibers marked as *SKY* in the fibers configuration. The RSS with sky subtracted is returned ini the field *final_rss* of the recipe result.

Then, the centroid of each fiducial object, marked as *REFERENCE* in the fibers configuration, is computed. The offset and rotation needed to center each fiducial object in its bundle is computed and returned

**class AcquireMOSRecipeInput**(**args*, ***kwds*)

AcquireMOSRecipeInput documentation.

**Attributes**

**extraction_offset** [ListOfType, requirement, optional, default=[0.0]] Offset traces for extraction

**extraction_region** [ListOfType, requirement, optional, default=[1000, 3000]] Region used to compute a mean flux

**ignored_sky_bundles** [ListOfType, requirement, optional] Ignore these sky bundles

**master_bias** [MasterBias, requirement] Master BIAS image

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**master_dark** [MasterDark, requirement, optional] Master DARK image

**master_fiberflat** [MasterFiberFlat, requirement] Master fiber flat calibration

**master_sensitivity** [MasterSensitivity, requirement, optional] Master sensitivity for flux calibration

[22] https://numina.readthedocs.io/en/latest/reference/array.html#module-numina.array.offrot

**master_slitflat** [MasterSlitFlat, requirement, optional] Master slit flat calibration

**master_traces** [MultiType, requirement] Apertures information for extraction

**master_twilight** [MasterTwilightFlat, requirement, optional] Master twlight flat calibration

**master_wlcalib** [WavelengthCalibration, requirement] Wavelength calibration table

**obresult** [ObservationResultType, requirement] Observation Result

**reference_extinction** [ReferenceExtinctionTable, requirement, optional] Reference extinction

**relative_threshold** [float, requirement, optional, default=0.3] Threshold for peak detection

`class` **`AcquireMOSRecipeResult`**`(`**args, **kwds*`)`

AcquireMOSRecipeResult documentation.

**Attributes**

**final_rss** [ProcessedRSS, product]

**offset** [list, product]

**qc** [QualityControlProduct, product]

**reduced_image** [ProcessedFrame, product]

**reduced_rss** [ProcessedRSS, product]

**rotang** [float, product]

**`RecipeInput`**

alias of `numina.core.metaclass.AcquireMOSRecipeInput`

**`RecipeResult`**

alias of `numina.core.metaclass.AcquireMOSRecipeResult`

## 5.3 Scientific Modes

The observing modes described in this section are those intended for the acquisition of scientific data by the observer. Here we describe all possible scientific observations to be carried out either with one of the MEGARA IFUs or the Fiber MOS.

### 5.3.1 LCB IFU scientific observation

**Mode** LCB IFU scientific observation

**Usage** Online, Offline

**Key** MegaraLcbImage

**Recipe** *`LCBImageRecipe`*

**Recipe input** `LCBImageRecipeInput`

**Recipe result** `LCBImageRecipeResult`

This mode sequence includes the required actions to obtain scientifically-valid data with the LCB IFU instrument mode of MEGARA.

### Requirements

This mode requires having the focal-plane cover configured, the instrument shutter open, to configure the VPH mechanism to select the grating to be used, to set the instrument mode to LCB IFU, to move the focusing mechanism to the position pre-defined for the specific VPH of choice, and to expose a certain time and to readout the detector in a series of exposures, being this series the LCB IFU image set.

A pre-requisite for this observing mode is to have previously executed a "Fine acquisition with the LCB IFU" auxiliary observing mode (TBC depending on the system absolute positioning precision and the observer requirements).

As part of the MEGARA on-line quick-look software the images to be obtained by this observing mode should be processed and the spectra extracted so to produce a view of the field at a selectable wavelength within the wavelength range covered by the VPH of choice. As a number of Fiber MOS positioners are devoted to the measure of the sky background simultaneously with the LCB IFU observations the on-line quick-look software should be able to subtract the spectrum of the sky from the spectra in each IFU spaxel and from the maps generated above. This software should have information on the specific focal-plane cover configuration being used.

In the case of observing relatively extended targets (comparable in size or larger than the LCB IFU field of view) but a mapping observing mode is not required to be used a blank-sky image set should be obtained using the same instrumental configuration as for the science target.

### Products

The observer will obtain LCB IFU image sets as part of the routine scientific operation of the instrument. The observatory staff could also make use of this observing mode to verify the status of the instrument using any source different from a standard star. In the case of observing a standard star the calibration mode standard star with the LCB IFU could be used instead.

### Recipe, inputs and results

**class** `megaradrp.recipes.scientific.lcb.`**`LCBImageRecipe`**(**args*, ***kwargs*)

Process LCB images.

This recipe processes a set of images obtained in **LCB image** mode and returns the sky subtracted RSS.

**See also:**

*`megaradrp.recipes.scientific.mos.MOSImageRecipe`*

#### Notes

Images provided by *obresult* are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is not None), bias, dark current (if *master_dark* is not None) and slit-flat (if *master_slitflat* is not None).

Images thus corrected are the stacked using the median. The result of the combination is saved as an intermediate result, named 'reduced_image.fits'. This combined image is also returned in the field *reduced_image* of the recipe result.

The apertures in the 2D image are extracted, using the information in *master_traces* and resampled according to the wavelength calibration in *master_wlcalib*. Then is divided by the *master_fiberflat*. The resulting RSS is saved as an intermediate result named 'reduced_rss.fits'. This RSS is also returned in the field *reduced_rss* of the recipe result.

The sky is subtracted by combining the the fibers marked as *SKY* in the fibers configuration. The RSS with sky subtracted is returned in the field *final_rss* of the recipe result.

If a *master_sensitivity* is provided (optional), RSS products will be flux calibrated. If *reference_extinction* is provided (optional), *final_rss* and *reduced_rss* will be extinction corrected. Notice that *sky_rss* is not corrected from extinction.

**class ImageRecipeInput**(**args*, ***kwds*)
ImageRecipeInput documentation.

**Attributes**

**extraction_offset** [ListOfType, requirement, optional, default=[0.0]] Offset traces for extraction

**ignored_sky_bundles** [ListOfType, requirement, optional] Ignore these sky bundles

**master_bias** [MasterBias, requirement] Master BIAS image

**master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask

**master_dark** [MasterDark, requirement, optional] Master DARK image

**master_fiberflat** [MasterFiberFlat, requirement] Master fiber flat calibration

**master_sensitivity** [MasterSensitivity, requirement, optional] Master sensitivity for flux calibration

**master_slitflat** [MasterSlitFlat, requirement, optional] Master slit flat calibration

**master_traces** [MultiType, requirement] Apertures information for extraction

**master_twilight** [MasterTwilightFlat, requirement, optional] Master twlight flat calibration

**master_wlcalib** [WavelengthCalibration, requirement] Wavelength calibration table

**obresult** [ObservationResultType, requirement] Observation Result

**reference_extinction** [ReferenceExtinctionTable, requirement, optional] Reference extinction

**relative_threshold** [float, requirement, optional, default=0.3] Threshold for peak detection

**class LCBImageRecipeResult**(**args*, ***kwds*)
LCBImageRecipeResult documentation.

**Attributes**

**final_rss** [ProcessedRSS, product]

**qc** [QualityControlProduct, product]

**reduced_image** [ProcessedFrame, product]

**reduced_rss** [ProcessedRSS, product]

**sky_rss** [ProcessedRSS, product]

**RecipeInput**
alias of `numina.core.metaclass.ImageRecipeInput`

**RecipeResult**
alias of `numina.core.metaclass.LCBImageRecipeResult`

### 5.3.2 Fiber MOS scientific observation

**Mode** Fiber MOS scientific observation

**Usage** Online, Offline

**Key** MegaraMosImage

**Recipe** *MOSImageRecipe*

**Recipe input** `MOSImageRecipeInput`

**Recipe result** `MOSImageRecipeResult`

This mode sequence includes the required actions to observe a list of targets with known celestial coordinates with MEGARA using the Fiber MOS instrument mode. The information on the assignment of targets and positioners is included in the form of a set of input catalogues generated off-line by the MEGARA Observing Preparation Software Suite (MOPSS). The reference position for each positioner is the center of the central fiber of the 7-fiber minibundle. This observing mode could be run with one of the sides of the focal-plane cover closed in order to reduce the cross-talk between positioners that would be placed adjacent in the pseudo-slit. Thus, the input catalogues should be specific of the focal-plane cover configuration to be used and both the on-line quick-look software and the off-line pipeline should include that information as a parameter or set of parameters. Note that the Fiber MOS catalogues and configuration files could be designed for their use under any focal-plane cover configuration. Even in that case the data processing software should know under which configuration a given image set was obtained.

### Requirements

This observing mode requires having the focal-plane cover configured, the instrument shutter open, to configure the VPH mechanism to select the grating to be used, to set the instrument mode to Fiber MOS, to move the focusing mechanism to the position pre-defined for the specific VPH of choice, to move all robotic positioners with a target associated in the input catalogues to the position of the corresponding target (these include science targets, reference stars for fine acquisition and positioners devoted to blank-sky measurements) and to expose a certain time and to readout the detector in a series of exposures, being this series the Fiber MOS image set.

A pre-requisite for running this observing mode is to have previously executed a "Fine acquisition with the Fiber MOS" auxiliary observing mode on the same field.

As part of the MEGARA on-line quick-look software, the image (or images) obtained should be processed and the spectra extracted. The observer might define a number of positioners to be placed on blank-sky regions of the field in order to improve sky subtraction. Alternatively, the user can also define a blank sky position. This is particularly important when observing individual stars in a nearby (Local Group) galaxy, for example, where the emission from the host galaxy is expected to contaminate even the outermost positioners. Should that be the case, the on-line quick-look software should be able to derive a sky spectrum from the blank-sky observation (if present) or the spectra of these positioners (if defined and no blank-sky observation is available) and subtract it from the spectra of the targets. The processed spectra should then be visualized using the on-line quick-look software. If neither a blank-sky observation nor blank-sky positioners are available no sky subtraction will be performed.

### Products

The observer will obtain Fiber MOS image sets as part of the routine scientific operation of the instrument. The observatory staff could also make use of this observing mode to verify the status of the instrument.

### Recipe, inputs and results

**class** `megaradrp.recipes.scientific.mos.`**`MOSImageRecipe`**(**args*, ***kwargs*)

Process MOS images.

This recipe processes a set of images obtained in **MOS image** mode and returns the sky subtracted RSS.

**See also:**

*megaradrp.recipes.scientific.lcb.LCBImageRecipe*

### Notes

Images provided by *obresult* are trimmed and corrected from overscan, bad pixel mask (if *master_bpm* is not None), bias, dark current (if *master_dark* is not None) and slit-flat (if *master_slitflat* is not None).

Images thus corrected are the stacked using the median. The result of the combination is saved as an intermediate result, named 'reduced_image.fits'. This combined image is also returned in the field *reduced_image* of the recipe result.

The apertures in the 2D image are extracted, using the information in *master_traces* and resampled according to the wavelength calibration in *master_wlcalib*. Then is divided by the *master_fiberflat*. The resulting RSS is saved as an intermediate result named 'reduced_rss.fits'. This RSS is also returned in the field *reduced_rss* of the recipe result.

The sky is subtracted by combining the the fibers marked as *SKY* in the fibers configuration. The RSS with sky subtracted is returned in the field *final_rss* of the recipe result.

If a *master_sensitivity* is provided (optional), RSS products will be flux calibrated. If *reference_extinction* is provided (optional), *final_rss* and *reduced_rss* will be extinction corrected. Notice that *sky_rss* is not corrected from extinction.

**class ImageRecipeInput**(**args*, ***kwds*)

ImageRecipeInput documentation.

**Attributes**

- **extraction_offset** [ListOfType, requirement, optional, default=[0.0]] Offset traces for extraction
- **ignored_sky_bundles** [ListOfType, requirement, optional] Ignore these sky bundles
- **master_bias** [MasterBias, requirement] Master BIAS image
- **master_bpm** [MasterBPM, requirement, optional] Master Bad Pixel Mask
- **master_dark** [MasterDark, requirement, optional] Master DARK image
- **master_fiberflat** [MasterFiberFlat, requirement] Master fiber flat calibration
- **master_sensitivity** [MasterSensitivity, requirement, optional] Master sensitivity for flux calibration
- **master_slitflat** [MasterSlitFlat, requirement, optional] Master slit flat calibration
- **master_traces** [MultiType, requirement] Apertures information for extraction
- **master_twilight** [MasterTwilightFlat, requirement, optional] Master twlight flat calibration
- **master_wlcalib** [WavelengthCalibration, requirement] Wavelength calibration table
- **obresult** [ObservationResultType, requirement] Observation Result
- **reference_extinction** [ReferenceExtinctionTable, requirement, optional] Reference extinction
- **relative_threshold** [float, requirement, optional, default=0.3] Threshold for peak detection

**class MOSImageRecipeResult**(**args*, ***kwds*)

MOSImageRecipeResult documentation.

**Attributes**

**final_rss** [ProcessedRSS, product]

**qc** [QualityControlProduct, product]

**reduced_image** [ProcessedFrame, product]

**reduced_rss** [ProcessedRSS, product]

**sky_rss** [ProcessedRSS, product]

**`RecipeInput`**
alias of `numina.core.metaclass.ImageRecipeInput`

**`RecipeResult`**
alias of `numina.core.metaclass.MOSImageRecipeResult`

## 5.4 Combined Modes

The observing modes described in this section are those that require the outputs of previous observing blocks.

### 5.4.1 Compute Sensitivity from Std Stars

**Mode**

**Usage** Offline

**Key** MegaraSensitivityStar

**Product** *`MasterSensitivity`*

**Recipe** *`Recipe`*

**Recipe input** `RecipeInput`

**Recipe result** `RecipeResult`

#### Recipe, inputs and results

**class** `megaradrp.recipes.combined.sensstar.`**`Recipe`**(**args, **kwargs*)
Process Sensitivity Star Recipe.

This recipe processes a set of images processed by the recipes of **Standard star with the FIBER MOS** or **Standard star with the LCB IFU** and returns the sensitivity correction required for flux calibration.

**See also:**

*`megaradrp.recipes.combined.extinctionstar.Recipe`*

**class** **`RecipeInput`**(**args, **kwds*)
RecipeInput documentation.

**Attributes**

**master_extinction** [ReferenceExtinctionTable, requirement] Atmospheric extinction

**obresult** [ObservationResultType, requirement] Observation Result

**reference_spectra** [ListOfType, requirement] Reference spectra of Std stars

**class** **`RecipeResult`**(**args, **kwds*)
RecipeResult documentation.

**Attributes**

**master_sensitivity** [MasterSensitivity, product]

**qc** [QualityControlProduct, product]

### 5.4.2 Compute Extinction and Sensitivity from Std Stars

**Mode** Compute Extinction from Std Stars

**Usage** Offline

**Key** MegaraExtinctionStar

**Product** *`MasterSensitivity`*, `Extinction`

**Recipe** *`Recipe`*

**Recipe input** `RecipeInput`

**Recipe result** `RecipeResult`

#### Recipe, inputs and results

**class** `megaradrp.recipes.combined.extinctionstar.`**`Recipe`**(**args*, ***kwargs*)

Process Sensitivity Star Recipe.

This recipe processes a set of images processed by the recipes of **Standard star with the FIBER MOS** or **Standard star with the LCB IFU** and returns the sensitivity correction and the atmospheric extinction required for flux calibration.

**See also:**

*`megaradrp.recipes.combined.sensstar.Recipe`*

**class** **`RecipeInput`**(**args*, ***kwds*)

RecipeInput documentation.

**Attributes**

**obresult** [ObservationResultType, requirement] Observation Result

**reference_spectra** [ListOfType, requirement] Reference spectra of Std stars

**class** **`RecipeResult`**(**args*, ***kwds*)

RecipeResult documentation.

**Attributes**

**master_extinction** [ReferenceExtinctionTable, product] Atmospheric extinction

**master_sensitivity** [MasterSensitivity, product]

**qc** [QualityControlProduct, product]

CHAPTER 6

# Data Products

Each recipe of the MEGARA Pipeline produces a set of predefined results, known as *data products*. In turn, the recipes may request different data products as computing requirements, effectively chaining the recipes.

For example, the requirements of `FiberFlatRecipe` include a `MasterDark` object. This object is produced by the recipe `DarkRecipe`, which in turn requires a `MasterBias` object.

## 6.1 FITS Keywords

The FITS keywords used by MEGARA are described in full detail elsewhere (document TEC/MEG/146).

In the following sections, we describe the keywords that are used by the pipeline.

### 6.1.1 Primary header

| Type | Keyword | Example | Explanation |
|---|---|---|---|
| L | SIMPLE | T | Standard FITS format |
| I | BITPIX | 16 | One of -64,-32,8,16,32 |
| I | NAXIS | 2 | # of axes in frame |
| I | NAXIS1 | 2048 | # of pixels per row |
| I | NAXIS2 | 2048 | # of rows |
| S | ORIGIN | 'GTC' | FITS originator |
| S | OBSERVAT | 'ORM' | Observatory |
| S | TELESCOP | 'GTC' | The telescope |
| S | INSTRUME | 'MEGARA' | The instrument |
| S | OBJECT | 'NGC 4594' | Target designation |
| S | OBSERVER | 'OBSERVER' | Who adquired the data |
| S | DATE-OBS | '2012-09-20T12:00:11.50' | Date of the start of the observation |
| S | DATE | '2012-09-20T12:14:12.78' | Date the file was written |

### 6.1.2 Required by the pipeline

| Type | Keyword | Example | Explanation |
|---|---|---|---|
| R | AIRMASS | 1.1908 | Mean airmass of the observation |
| R | MJD-OBS | 72343.34324 | Modified JD of the start of the observation |
| S | IMAGETYP | 'FLAT' | Type of the image |
| S | VPH | 'LR-R' | Type of VPH |
| S | OBSTYPE | 'SLITFLAT' | Type of observation |
| R | EXPOSED | | Exposure time in seconds |
| R | EXPTIME | | Exposure time in seconds (synonim) |
| R | DARKTIME | | TBD |
| S | OBSMODE | 'SLITFLAT | Identifier of the observing mode |

### 6.1.3 FIBERS extension

The state of the focal plane of MEGARA is stored in a dedicated extension names *FIBERS*. This extension contains only headers, the data part will be empty.

| Type | Keyword | Example | Explanation |
|---|---|---|---|
| I | NFIBERS | 643 | Number of fibers |
| I | NSPAXEL | 644 | Number of spaxels |
| I | NBUN-DLES | 92 | Number of fiber bundles |
| S | INSMODE | LCB | Name of active pseudo slit |
| S | CONFID | 'b7d35e7df0274fde..' | Unique identificator of the configuration |
| I | BUNnnn_P | 0 | Priority of the target in this bundle |
| S | BUNnnn_I | 'unknown ' | Name of the target |
| S | BUNnnn_T | 'UNASSIGNED' | Type of target ('STAR', 'SKY', 'TARGET', 'UNAS-SIGNED' |
| I | FIBmmm_B | nnn | ID of the bundle |
| F | FIB-mmm_D | +3.34565 | Declination of the spaxel |
| F | FIB-mmm_R | 12.342223 | Right Ascension of the spaxel |
| F | FIB-mmm_O | 0.0 | Position Angle of the Fiber |
| L | FIB-mmm_A | T | Is fiber active? |
| F | FIB-mmm_X | -0.4646226291303512 | X position of the fiber in the focal plane |
| F | FIB-mmm_Y | 63.63025 | Y position of the fiber in the focal plane |

## 6.2 Data products

These data products are saved to disk as FITS files. MEGARA DRP makes use of the FITS headers to record information about the data processing. This information may be recorded using other methods as well, such as the GTC Database.

The following headers are included in all image data products and record information about the version of Numina and the name and version of the recipe used.

```
NUMXVER = '0.13.0    '            / Numina package version
NUMRNAM = 'BiasRecipe'            / Numina recipe name
NUMRVER = '0.1.0     '            / Numina recipe version
NUMTYP  = 'TARGET    '            / Data product type
```

HISTORY keywords may be used also, but the information in these keyword may not be easily indexed.

### 6.2.1 Generic types

#### Processed Frame

Processed Frame is the type of any image produced by the pipeline that represents a view of the detector. Its size may be 4096x4112 for trimmed images or 4196x4212 for unprocessed, raw images.

Processed Frame is represented by *ProcessedFrame*.

#### Processed RSS

Processed Row Stacked Spectra is the type of any image produced by the pipeline that represents a view of the focal plane, using extracted fibers. It will have 623 rows in LCB mode and 644 rows in MOS mode, each row representing the extracted spectrum of one fiber. The number of columns will be 4112 or larger, depending on the stage of the reduction.

Procesed RSS images will tipically have a *FIBERS* extension.

Processed RSS is represented by *ProcessedRSS*.

#### Processed Spectrum

Processed spectrum is the type of any image produced by the pipeline that represents the spectrum of one object. Its data content will be a 1D array.

Processed Spectrum is represented by *ProcessedSpectrum*.

### 6.2.2 Calibrations

#### Master Bias frames

Bias frames are produced by the recipe *BiasRecipe*. Each bias frame is a multiextension FITS file with the following extensions.

| Extension name | Type | Version | Contents |
|---|---|---|---|
| PRIMARY | Primary | | The bias level |
| VARIANCE | Image | | Variance of the bias level |
| MAP | Image | | Number of pixels used to compute the bias level |

Master bias frames are represented by *MasterBias*.

### Master Dark frames

Master dark frames are produced by the recipe `DarkRecipe`. Each dark frame is a multiextension FITS file with the following extensions.

| Extension name | Type | Version | Contents |
|---|---|---|---|
| `PRIMARY` | Primary | | The dark level |
| `VARIANCE` | Image | | Variance of the dark level |
| `MAP` | Image | | Number of pixels used to compute the dark level |

Master dark frames are represented by `MasterDark`.

### Master Bad Pixel Mask

Master Bad Pixel Mask is produced by the recipe `BadPixelsMaskRecipe`. Each bad pixel mask frame is a multiextension FITS file with the following extensions.

| Extension name | Type | Version | Contents |
|---|---|---|---|
| `PRIMARY` | Primary | | The Bad Pixel Mask level |

Master bad pixel mask frames are represented by `MasterBPM`.

### Master Slit Flat

Master Slit Flat is produced by the recipe `SlitFlatRecipe`. Each slit flat frame is a multiextension FITS file with the following extensions.

| Extension name | Type | Contents |
|---|---|---|
| `PRIMARY` | Primary | The Slit Flat level |

Masterslit flat frames are represented by `MasterSlitFlat`.

### Master Traces

Master Fiber Flat is produced by the recipe `TraceMapRecipe`. The result is a JSON[23] file where each one of the records belongs to a given fiber in the RSS file. Moreover, each one of the records has the next information:

| Field | Type | Contents |
|---|---|---|
| `boxid` | Integer | Number of the box |
| `fibid` | Integer | Number of the fiber |
| `fitparms` | Primary | Polyfit algorithm result |
| `start` | Integer | X-Coordenate in the Flat image |
| `stop` | Integer | X-Coordenate in the Flat image |

In the following, a real example of the fourth fiber which is in the first box can be seen in the yaml format:

[23] http://www.json.org/

```
- boxid: 1
  fibid: 4
  fitparms: [2.6909627476636523e-18, -3.0949058966515047e-14, 1.872326137294402e-
↪10,1.1602592442769502e-06, -0.0009443161994027746, 262.01840282676613]
  start: 4
  stop: 3594
```

Master Tracemap files are represented by `TraceMap`.

### Master Wavelength Calibration

Master wavelength calibration is produced by the recipe `ArcCalibrationRecipe`. The result is a JSON[24] file where each one of the records belongs to a given fiber in the RSS file. Moreover, each one of the records or `apertures` has the next fields:

| Field | Type | Contents |
|---|---|---|
| `features` | List | List with the arc's information |
| `function` | Dictionary | Number of pixels used to compute the dark level |
| `id` | Integer | Number the corresponding fiber |

Additionally, each one of the elements that belongs to the `features` corresponds to each one of the arc lines that has been found in the RSS image. The dictionary that each element has, contains the next information:

| Field | Type | Contents |
|---|---|---|
| `category` | String | Type of the arc |
| `flux` | Float | Flux of the arc |
| `fwhm` | Float | Full Width at Half Maximum of the arc |
| `reference` | Float | Line in the Catalog lines |
| `wavelength` | Float | Predicted line |
| `xpos` | Float | X-coordenate of the arc in the RSS image |
| `ypos` | Float | Y-coordenate of the arc in the RSS image |

Finally, the `function` dictionary has three elements: `coefficients`, `method` and `order` fields. Coefficients has the result of executing the `polynomial.polyfit` numpy method. Method field has the name of the algorithm used. Order field has the polynomial degree.

In the following, an example of the first fiber of a real JSON file with only two arc lines can be seen:

```
{
  "aperture": {
    "features": [
      {
        "category": "E",
        "flux": 50212.563405324945,
        "fwhm": 3.438967092459162,
        "reference": 6013.2816999999995,
        "wavelength": 6013.2847301957181,
        "xpos": 33.267395825699928,
        "ypos": 251.10097403866305
      },
    ],
    "function": {
      "coefficients": [6001.573165443434,0.35298729563735487,-2.898410563853586e-
↪05,1.858317850662985e-08,-8.411429549924489e-12,1.4341696725726076e-15],
      "method": "least squares",
```

(continues on next page)

[24] http://www.json.org/

(continued from previous page)

```
        "order": 5
    },
    "id": 2
}
```

Master Wavelength calibration file is represented by *WavelengthCalibration*.

### Master Fiber Flat

Master Fiber Flat is produced by the recipe *FiberFlatRecipe*. Each master fiber flat frame is a multiextension FITS file with the following extensions.

| Extension name | Type | Version | Contents |
|---|---|---|---|
| PRIMARY | Primary | | The Fiber Flat level |
| FIBERS | Image | | Description of the focal plane |

Master fiber flats frames are represented by *MasterFiberFlat*.

### Master Twilight Flat

Master Twilight Flat is produced by the recipe *TwilightFiberFlatRecipe*. Each twilight flat frame is a multiextension FITS file with the following extensions.

| Extension name | Type | Version | Contents |
|---|---|---|---|
| PRIMARY | Primary | | The Twilight Flat level |
| FIBERS | Image | | Description of the focal plane |

Master twilight flat frames are represented by *MasterTwilightFlat*.

### Master Sensitivity

Master sensitivity star image is produced by the recipe `Recipe`.

| Extension name | Type | Version | Contents |
|---|---|---|---|
| PRIMARY | Primary | | The Sensitivity Star Image level |

Master sensitivity star image is represented by *MasterSensitivity*.

### Master Extinction

Master extinction star image is produced by the recipe `Recipe`.

| Extension name | Type | Version | Contents |
|---|---|---|---|
| PRIMARY | Primary | | The Extinction Star Image level |

Master extinction star image is represented by `Extinction`.

## 6.2.3 Reference calibrations

The following types represent types used for calibration, but that are not the result of any recipe. Examples of this type are the spectra of flux standars or the tables of spectral lines of calibration lamps.

### Reference Spectrum

A tabular representation of the spectral energy distribution of a standard star. The first column contains wavelength (in Angstroms) and the second column the flux in erg/s/cm^2/Angstrom

Reference spectrum is represented by `ReferenceSpectrum`.

# CHAPTER 7

# Reduction Recipes

## 7.1 Execution environment of the Recipes

Recipes have different execution environments. Some recipes are designed to process observing modes required while observing at the telescope. These modes are related to visualization, acquisition and focusing. The corresponding Recipes are integrated in the GTC environment. We call these recipes the **Data Factory Pipeline**, (DFP).

Other group of recipes are devoted to scientific observing modes and auxiliary calibrations. These Recipes constitute the **Data Reduction Pipeline**, (DRP). The software is meant to be standalone, users shall download the software and run it in their own computers, with reduction parameters and calibrations provided by the instrument team.

Users of the DRP may use the simple Numina CLI (Command Line Interface). Users of the DFP shall interact with the software through the GTC Inspector.

## 7.2 Recipe Parameters

MEGARA Recipes based on Numina have a list of required parameters needed to configure the Recipe properly. The Recipe announces the required parameters with the following syntax (the syntax is subject to changes).

```
class SomeRecipeInput(RecipeInput):
    master_dark = DataProductParameter(MasterDark, 'Master dark image')
    some_numeric_value = Parameter(0.45, 'Some numeric value'),

@define_input(SomeRecipeInput)
class SomeRecipe(RecipeBase):
    ...
```

When the Recipe is configured properly, it is executed with an observing block data structure as input. When is run using Numina CLI, this data structure is created from an user-provided text file. The recipe requirements values are either provided in a text file or have default values.

## 7.3 Recipe Products

Recipes based on Numina provide a list of products created by the recipe. The Recipe announces the required parameters with the following syntax (the syntax is subject to changes).

```
class SomeRecipeInput(RecipeInput):
    master_dark = DataProductParameter(MasterDark, 'Master dark image')
    some_numeric_value = Parameter(0.45, 'Some numeric value'),

class SomeRecipeResult(RecipeResult):
    master_flat = Product(MasterDark)

@define_input(SomeRecipeInput)
@define_result(SomeRecipeResult)
class SomeRecipe(RecipeBase):
    ...
```

The data products of the MEGARA DRP are describe in *Data Products*

# CHAPTER 8

# Reference

**Release** 0.8

**Date** Dec 05, 2018

**Warning:** This "Reference" is still a work in progress; some of the material is not organized, and several aspects of MEGARA DRP are not yet covered sufficient detail.

## 8.1 `megaradrp.core` — Base classes for processing

**class** megaradrp.core.recipe.**MegaraBaseRecipe**(**args*, ***kwargs*)

Base clase for all MEGARA Recipes

**Parameters**

**intermediate_results** [bool, optional] If True, save intermediate results of the Recipe

**Attributes**

**obresult** [ObservationResult, requirement]

**qc** [QualityControl, result, QC.GOOD by default]

**logger :** recipe logger

**datamodel** [MegaraDataModel]

**class MegaraBaseRecipeInput**(**args*, ***kwds*)

MegaraBaseRecipeInput documentation.

**Attributes**

**obresult** [ObservationResultType, requirement] Observation Result

**attrs**()

**obresult**

The Recipe requires the result of an observation.

**classmethod stored**()

**validate**()
Validate myself.

class **MegaraBaseRecipeResult**(**args, **kwds*)
MegaraBaseRecipeResult documentation.

**Attributes**

**qc** [QualityControlProduct, product]

**attrs**()

**qc**
Product holder for RecipeResult.

Deprecated since version 0.16: *Product* is replaced by *Result*. It will be removed in 1.0

**store_to**(*where*)

classmethod **stored**()

**validate**()
Validate myself.

**RecipeInput**
alias of `numina.core.metaclass.MegaraBaseRecipeInput`

**RecipeResult**
alias of `numina.core.metaclass.MegaraBaseRecipeResult`

**build_recipe_input**(*ob, dal*)
Build a RecipeInput object.

**configure**(***kwds*)

static **create_default_runinfo**()

classmethod **create_input**(**args, **kwds*)
Pass the result arguments to the RecipeInput constructor

classmethod **create_result**(**args, **kwds*)
Pass the result arguments to the RecipeResult constructor

**datamodel = <megaradrp.datamodel.MegaraDataModel object>**

**gather_info**(*recipeinput*)

**get_filters**()

**init_filters**(*rinput, ins*)

**init_filters_generic**(*rinput, getters, ins*)

**logger = <logging.Logger object>**

**obsres_extractor**(*obsres, tag_keys*)

classmethod **products**()

classmethod **requirements**()

**run**(*recipe_input*)

**run_qc**(*recipe_input, recipe_result*)
Run Quality Control checks.

**save_intermediate_array**(*array, name*)
Save intermediate array object as FITS.

**save_intermediate_img**(*img, name*)
Save intermediate FITS objects.

**save_structured_as_json**(*structured, name*)

**set_base_headers**(*hdr*)
Set metadata in FITS headers.

**types_getter**()

**validate_input**(*recipe_input*)
Method to customize recipe input validation.

**See also:**

`numina.core.validator.validate`[25]

**validate_result**(*recipe_result*)
Validate the result of the recipe

## 8.2 `megaradrp.instrument` — Static configuration

**class** `megaradrp.instrument.loader.`**Loader**
Instrument configuration loader for MEGARA

## 8.3 `megaradrp.processing` — Processing functions

## 8.4 `megaradrp.processing.wavecalibration` —

Corrector for wavecalibration

**class** `megaradrp.processing.wavecalibration.`**WavelengthCalibrator**(*solutionwl, datamodel=None, dtype='float32'*)
A Node that applies wavelength calibration.

**map_borders**(*wls*)
Compute borders of pixels for interpolation.

The border of the pixel is assumed to be midway of the wls

## 8.5 `megaradrp.products` — Data products of the MEGARA pipeline

**class** `megaradrp.types.`**MegaraFrame**(**args, **kwds*)
A processed frame

**class** `megaradrp.types.`**ProcessedFrame**(**args, **kwds*)
A processed frame

**class** `megaradrp.types.`**ProcessedImage**(**args, **kwds*)
A processed image

**class** `megaradrp.types.`**ProcessedRSS**(**args, **kwds*)
A processed RSS image

**class** `megaradrp.types.`**ProcessedMultiRSS**(**args, **kwds*)
A processed RSS image not to be stored

**class** `megaradrp.types.`**ProcessedSpectrum**(**args, **kwds*)
A 1d spectrum

[25] https://numina.readthedocs.io/en/latest/reference/core.html#numina.core.validator.validate

**class** megaradrp.types.**ProcessedImageProduct**(**args, **kwargs*)

**class** megaradrp.types.**ProcessedRSSProduct**(**args, **kwargs*)

**class** megaradrp.types.**ProcessedSpectrumProduct**(**args, **kwargs*)

**class** megaradrp.types.**MasterBPM**(**args, **kwargs*)
Bad Pixel Mask product

**class** megaradrp.types.**MasterBias**(**args, **kwargs*)
A Master Bias image

**class** megaradrp.types.**MasterDark**(**args, **kwargs*)
A Master Dark image

**class** megaradrp.types.**MasterSlitFlat**(**args, **kwargs*)

**class** megaradrp.types.**MasterFiberFlat**(**args, **kwargs*)

**class** megaradrp.types.**MasterTwilightFlat**(**args, **kwargs*)

**class** megaradrp.products.structured.**BaseStructuredCalibration**(*instrument='unknown'*)

**class** megaradrp.products.tracemap.**TraceMap**(*instrument='unknown'*)
Trace map calibration product

**class** megaradrp.products.modelmap.**ModelMap**(*instrument='unknown'*)

**class** megaradrp.products.wavecalibration.**WavelengthCalibration**(*instrument='unknown'*)
Wavelength Calibration Product

**class** megaradrp.types.**MasterSensitivity**(**args, **kwargs*)
Sensitivity correction.

**class** megaradrp.types.**ReferenceExtinctionTable**(**args, **kwargs*)
Atmospheric Extinction.

**class** megaradrp.types.**ReferenceSpectrumTable**(**args, **kwargs*)
The spectrum of a reference star

## 8.6 megaradrp.recipes — Reduction Recipes for MEGARA

## 8.7 megaradrp.types — MEGARA data types

## 8.8 megaradrp.datamodel — MEGARA datamodel

Data model for MEGARA

**class** megaradrp.datamodel.**BundleConf**
Description of a bundle

**class** megaradrp.datamodel.**FiberConf**
Description of the fiber

**class** megaradrp.datamodel.**FibersConf**
Global configuration of the fibers

**active_fibers**()

**bundles_to_table**()
Convert bundles to a Table

**conected_fibers**(*valid_only=False*)

**fibers_to_table**()
Convert fibers to a Table

**inactive_fibers**()

**invalid_fibers**()

**sky_fibers**(*valid_only=False, ignored_bundles=None*)

**spectral_coverage**()

**valid_fibers**()

**class** megaradrp.datamodel.**MegaraDataModel**
Data model of MEGARA images

**PLATESCALE = 1.12**

**db_info_keys = ['instrument', 'object', 'observation_date', 'uuid', 'type', 'mode',**

**db_info_keys_extra = ['vph', 'insmode']**

**fiber_scale_unit**(*img, unit=False*)

**gather_info_oresult**(*val*)

**get_fiberconf**(*img*)
Obtain FiberConf from image

**get_fiberconf_default**(*insmode*)
Obtain default FiberConf object

**get_imgid**(*img*)
Obtain a unique identifier of the image.

**Parameters**

**img** [astropy.io.fits.HDUList]

**Returns**

**str:** Identification of the image

**meta_dinfo_headers = ['exptime', 'observation_date', 'vph', 'vphpos', 'insmode', 'fo**

**meta_info_headers = ['instrument', 'object', 'observation_date', 'uuid', 'type', 'mo**

**query_attrs = {'insconf': <megaradrp.datamodel.QueryAttribute object at 0x7f409b69**

**class** megaradrp.datamodel.**QueryAttribute**(*name, tipo, description=''*)

**class** megaradrp.datamodel.**TargetType**
Possible targest in a fiber bundle

**BLANK = 4**

**REFERENCE = 5**

**SKY = 4**

**SOURCE = 1**

**STAR = 5**

**UNASSIGNED = 3**

**UNKNOWN = 2**

megaradrp.datamodel.**read_fibers_extension**(*hdr, insmode='LCB'*)
Read the FIBERS extension

**Parameters**

**hdr:** FITS header

**insmode: str** default INSMODE

**Returns**

**FibersConf**

## 8.9 `megaradrp.utils` — MEGARA utilities

Some utils

`megaradrp.utils.`**`add_collapsed_mos_extension`**(*img*, *size=7*, *axis=0*)
Add a collapsed image extension

**Parameters**

**img: astropy.io.fits.HDUList**

**Returns**

**astropy.io.fits.HDUList** Updated image

## 8.10 `megaradrp.validators` — MEGARA validators

Validators for Observing modes

`megaradrp.validators.`**`validate_focus`**(*obresult*)
Validate FOCUS_SPECTROGRAPH

## 8.11 `megaradrp.visualization` — MEGARA visualization

`megaradrp.visualization.`**`hexplot`**(*axis*, *x*, *y*, *z*, *scale=1.0*, *extent=None*, *cmap=None*, *norm=None*, *vmin=None*, *vmax=None*, *alpha=None*, *linewidths=None*, *edgecolors='none'*, ***kwargs*)
Make a hexagonal grid plot.

**Returns**

**object: matplotlib.collections.PolyCollection**

## 8.12 `megaradrp.simulation` — Simulation modules

Sequences for observing modes of MEGARA

`megaradrp.simulation.actions.`**`add_targets_lcb2`**(*targets*, *subfinal*, *wl*, *fibrad*, *pos_x*, *pos_y*, *oe*, *telescope*, *atmosphere*)
This correction includes DAR

`megaradrp.simulation.actions.`**`simulate_bias`**(*detector*)
Simulate a BIAS array.

`megaradrp.simulation.actions.`**`simulate_dark`**(*detector*, *exposure*)
Simulate a DARK array,

`megaradrp.simulation.actions.`**`simulate_dark_fits`**(*factory*, *instrument*, *exposure*, *repeat=1*)
Simulate a DARK FITS.

`megaradrp.simulation.actions.`**`simulate_flat`**(*detector*, *exposure*, *source*)
Simulate a FLAT array,

**class** `megaradrp.simulation.control.`**`ControlSystem`**(*factory*)
Top level

**class** megaradrp.simulation.convolution.**HexagonA**(*amplitude=1, x_0=0, y_0=0, radius=1, angle=0, **kwargs*)

**static evaluate**(*x, y, amplitude, x_0, y_0, radius, angle*)
Evaluate the model on some input variables.

**class** megaradrp.simulation.cover.**MegaraCover**(*parent=None*)
MEGARA Cover

**set_mode**(*mode*)
Cover in the focal plane.

**class** megaradrp.simulation.detector.**MegaraDetector**(*name, shape, oscan, pscan, qe=1.0, qe_wl=None, dark=0.0, readpars1=None, readpars2=None, bins='11', direction='normal'*)
Simple MEGARA detector.

**init_regions**(*detshape, oscan, pscan, bng*)
Create a image with overscan for testing.

**saturate**(*x*)
Compute non-linearity and saturation.

**class** megaradrp.simulation.detector.**MegaraDetectorSat**(*name, shape, oscan, pscan, qe=1.0, qe_wl=None, dark=0.0, readpars1=None, readpars2=None, bins='11', direction='normal'*)

**saturate**(*x*)
Compute non-linearity and saturation.

**class** megaradrp.simulation.detector.**ReadParams**(*gain=1.0, ron=2.0, bias=1000.0*)
Readout parameters of each channel.

megaradrp.simulation.detector.**binning**(*arr, br, bc*)
Return a binned view if 'arr'

**class** megaradrp.simulation.efficiency.**InterpolFitsUVES**(*fname, fill_value=0.0*)
Interpolate spectrum in UVES format.

This is the format of the sky spectrum file

Extended multiwavelength simulation Simple monocromatic simulation

megaradrp.simulation.instrument.**compute_kernel**(*sigma, d=0.5, truncate=5.0*)
A kernel representing a Gaussian convolved with a square.

megaradrp.simulation.instrument.**pixcont_int**(*i, x0, sig*)
Integrate a gaussian profile.

megaradrp.simulation.instrument.**pixcont_int_pix**(*i, x0, sig, d=0.5*)
Integrate a gaussian profile.

Computing differential atmospheric refraction

megaradrp.simulation.refraction.**differential_p**(*zenith_distance, wl, wl_reference, temperature, pressure, relative_humidity*)
Differential refraction as given by 1982PASP…94..715F

## 8.13 Indices and tables

- genindex
- modindex
- search
- *Glossary*

# CHAPTER 9

# Glossary

**DFP** Data Factory Pipeline

**DRP** Data Reduction Pipeline

**observing mode** One of the prescribed ways of observing with an instrument

**recipe** A software object that processes the data obtained with a given observing mode of the instrument

# CHAPTER 10

# Cookbook

**Release** 0.8

**Date** Dec 05, 2018

# Python Module Index

# Index