Python snippets

In order to edit a set of input files and make them ready for the execution of our tasks, we must add the following keywords to the header of each FITS file.

CARD Name	***Value***
INSTRUME	MICADO
MJD-OBS	timestamp or obs date
ESO DPR CATG	CALIB
ESO DPR TYPE	<category>
ESO DPR TECH	TABLE or IMAGE

This code snippet in Python can be useful for the purpose:

```
def create_output_img(filename, catg, tech):
    with fits.open(filename) as hdul:
        hdul[0].header["INSTRUME"] = "MICADO"
        hdul[0].header["MJD-OBS"] = Time(datetime.now().isoformat()).mjd
        hdul[0].header["ESO DPR CATG"] = "CALIB"
        hdul[0].header["ESO DPR TYPE"] = catg
        hdul[0].header["ESO DPR TECH"] = tech
        hdul.writeto(f"./{catg.upper()}.fits", overwrite=True)
```

MCD_PSFR_SCAO_OTF_MICADO recipe - Input data and Python Code

Concerning PUP_VIEWS_RAW, please refer to documentation provided with the **mcd_psfr_scao_otf_par** recipe. The dataset and the necessary python script are described at the end of the document.

For the other input files, you can use the following Python code. The code will create empty FITS files (no data, only header) and add the necessary information to each header. Then the files are saved in the destination folder (where the script is executed).

```
# prepare_micado_inputs.py
from pathlib import Path
import numpy as np
from astropy.io import fits
from astropy.time import Time
from datetime import datetime
```

```
def create_empty_files(catg, tech, pro=False):
    print(f"    add {catg} and {tech}...")
    input_path=Path(".")
    hh = fits.PrimaryHDU(data=None)
    hh.header["INSTRUME"] = "MICADO"
    hh.header["MJD-OBS"] = Time(datetime.now().isoformat()).mjd
    if pro:
        hh.header["ESO PRO CATG"] = catg
    else:
        hh.header["ESO DPR CATG"] = "CALIB"
        hh.header["ESO DPR TYPE"] = catg
        hh.header["ESO DPR TECH"] = tech
    hh.writeto(f"./{catg.upper()}.fits", overwrite=True)


if __name__ == "__main__":
    categories=["FOCALPLANE", "OFFSET", "MASTER_DARK_IMG", "MASTER_FLAT_IMG" ]
    techs = ["IMAGE", "TABLE", "IMAGE", "IMAGE"]
    for c,t in zip(categories, techs): # create other data
         create_empty_files(c, t, pro=c.startswith("MASTER"))
```

MCD_PSFR_SCAO_OTF_PAR recipe - Input data and Python Code

You can use the following Python code to set up your input files contained in this archive: https://drive.google.com/file/d/1EJ7_glM4wVJzv2qrCB_7BR_BUtfixFun/view?usp=drive_link
The code opens each file and adds the necessary information to each header. Then the data (and modified header) is saved in the destination folder (where the script is executed).

Remark: The following code takes into account the input files for this recipe. Consider also the python script described in documentation of **mcd_psfr_scao_otf_micado** recipe for the complete dataset.

```
# prepare_inputs.py
from pathlib import Path
import numpy as np
from astropy.io import fits
from astropy.time import Time
from datetime import datetime

def create_output_img(filename, catg, tech):
    input_path=Path("./") # adjust this with your path!
    with fits.open(input_path / filename) as hdul:
        print(f" -- FILE = {filename}")
        print(f"    add DPR.TYPE={catg} and DPR.TECH={tech}")
        hdul[0].header["INSTRUME"] = "MICADO"
        hdul[0].header["MJD-OBS"] = Time(datetime.now().isoformat()).mjd
        hdul[0].header["ESO DPR CATG"] = "CALIB"
        hdul[0].header["ESO DPR TYPE"] = catg
        hdul[0].header["ESO DPR TECH"] = tech
        #if catg== "PUPIL":
        #    hdul[0].header["PIXSIZE"] = 0.05025929565
```

```
        hdul.writeto(f"./{catg.upper()}.fits", overwrite=True)
        print(hdul.info())

if __name__ == "__main__":
    files = ["ifs_pupil.fits", "modal_gains.fits", "volts.fits", "basis.fits", "cmat.fits",
"cropped_M4IF.fits"]
    categories=["PUPIL", "MODAL_OPT", "DMCOMMAND", "BASISMATRIX", "CONTMATRIX", "REF_ELT_MODEL"
]
    techs = ["IMAGE", "TABLE", "TABLE", "TABLE", "TABLE", "TABLE"]
    for f,c,t in zip(files, categories, techs):
        create_output_img(f,c,t)
```

EDPS graph modification

Open your local EDPS package. In my case, since I use venv, it's placed in the local .venv folder: `.venv/lib/python3.13/site-packages/edps`
Navigate into the "generator" folder and open the file called "graph.py". Then modify the function "simple_graph" (line 117) as follows (in **bold**, modified lines):

```
    def simple_graph(self) -> str:
            template = self.environment.get_template(SIMPLE_GRAPH_TEMPLATE)
            raw_types = {ds.name for ds in self.data_sources if ds.is_main_input}
            static_calibs =  {ds.name for ds in self.data_sources if ds.is_assoc_input}
            tasks = {t.name for t in self.tasks if not t.subworkflow_name}
            subworkflows = {t.subworkflow_name[0] for t in self.tasks if t.subworkflow_name}
            edges = [(self.get_name_or_subworkflow(n0), self.get_name_or_subworkflow(n1))
                     for n0, n1 in self.graph.edges]
            edges = [(e0, e1) for e0, e1 in edges if e0 != e1]
            colors = [1, 11, 2, 10, 3, 9, 4, 8, 5, 7]
            edge_colors = {}
            for i, (n0, n1) in enumerate(edges):
                if n0 in raw_types:
                    edge_colors[n0] = colors[i % len(colors)]
                if n0 in static_calibs:
                    edge_colors[n0] = colors[i % len(colors)]
            context = {
                "workflow": self.workflow_name,
                "raw_types": raw_types,
                "static_calibs": static_calibs,
                "tasks": tasks,
                "subworkflows": subworkflows,
                "edges": edges,
                "edge_colors": edge_colors
            }
            return template.render(context)
```

Modify the template "simple_graph.dot" into the "templates" subfolder, adding the following code just after the first subgraph section (called cluster1):

```
    subgraph cluster2 {
            node [colorscheme=spectral11 fontsize=8 shape=box penwidth=4]
            fontcolor=goldenrod
            fontsize=30
            label="Static Calibrations"
```

```
    {% for name in static_calibs %}
    "{{ name }}" [color={{ edge_colors[name] }}]
    {% endfor %}
}
```

Remark: You must shutdown and restart EDPS to see the changes.