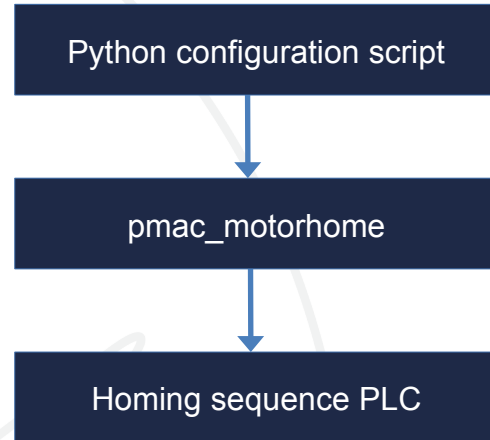


Overview

- Will be available on github
- Uses pipenv to manage dependencies
- Python script configures homing routines to be rendered
- Homing plc rendered from templates of code blocks (snippets)
- Can define custom routines
- Can add new snippets



Configuration script

- Python3 interface to configure homing sequences
- Context managers handle boiler plate code (e.g. rendering PLC code to file at the end)
- Can define custom sequences and routines, and add custom PLC code

```
from pmac_motorhome.commands import group, motor, plc
from pmac_motorhome.sequences import home_hsw

with plc(
    plc_num=12,
    controller="GeoBrick",
    filepath="/tmp/PLC12_SLITS1_HM.pmc",
):
    with group(group_num=3):
        motor(axis=1)
        motor(axis=2)

    home_hsw()
```

Rendering the homing routine

- Uses the Jinja2 templating engine
- Templates provide conditional logic and the use of loops
- Can include other templates, with options/arguments sent to them
- Snippets are small templates representing functional code blocks
- Allows python-like expressions to be evaluated and rendered into strings

Adding Functionality

- New snippets and sequences can be added to the source code
- Documentation includes:
 - Tutorials
 - How-to guides
 - Explanations for understanding
 - Reference material

- Will be available on github
- Uses pipenv to manage dependencies
- Runs in a virtual environment
- Can provide a lightweight-venv with which to run configuration script
- Extendable with new snippets, functions, homing sequences
- Documentation contains tutorials, how-to guides, explanations and reference material.

Python configuration script

pmac_motorhome

Homing sequence PLC

```
from pmac_motorhome.commands import group, motor, plc
from pmac_motorhome.sequences import home_hsw

with plc(
    plc_num=12,
    controller="GeoBrick",
    filepath="/tmp/PLC12_SLITS1_HM.pmc",
):
    with group(group_num=3):
        motor(axis=1)
        motor(axis=2)

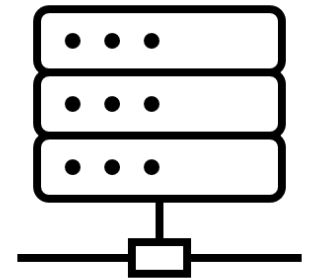
    home_hsw()
```

Simple configuration script to home two motors



PLCxx_HM.pmc

Upload
PLC code



Motion controller

Custom sequences can be defined in config script, or in pmac_motorhome source code

Context managers for plc and group objects

Add motors to a homing group with an axis number and jog distance

```
from pmac_motorhome.commands import PostHomeMove, group, motor, only_axes, plc
from pmac_motorhome.sequences import home_hsw
from pmac_motorhome.snippets import drive_to_limit

def custom_slits_hsw(posx, negx, posy, negy):
    drive_to_limit(homing_direction=False) # drive all slits to limit away from home

    with only_axes(posx, posy): # home and return to limit only positive slits
        home_hsw()
        drive_to_limit(homing_direction=False)

    with only_axes(negx, negy): # home and return to limit only negative slits
        home_hsw()
        drive_to_limit(homing_direction=False)

    with plc(
        plc_num=12,
        controller="GeoBrick",
        filepath="/tmp/PLC12_CUSTOM_SLITS_HM.pmc",
    ):
        initial = PostHomeMove.initial_position
        with group(group_num=2, post_home=initial):
            motor(axis=1, jdist=-400)
            motor(axis=2, jdist=-400)
            motor(axis=3, jdist=-400)
            motor(axis=4, jdist=-400)

        custom_slits_hsw(posx=1, negx=2, posy=3, negy=4)
```

Access to snippet functions to explicitly construct a homing sequence

only_axes context to specify axes for subset of axes in a group

Configuration script using drive_to_limit snippet function to define a custom homing routine for a group of 4 axes

Uses the Jinja2 templating engine

Templates provide conditional logic and the use of loops

Snippets are small templates representing functional code blocks

Can include other templates, with options/arguments sent to them

Allows python-like expressions to be evaluated and rendered into strings

```
{% if group.post and group.post != "" %}
{% include "debug_pause.pmc.jinja" %}

;---- PostHomeMove State ----
if (HomingStatus = StatusHoming or HomingStatus = StatusDebugHoming)
    HomingState=StatePostHomeMove
    ; Execute the move commands
    if (HomingStatus = StatusHoming or HomingStatus = StatusDebugHoming)
        {{ group.post }}
    endif
endif
endif

{% endif %}
```

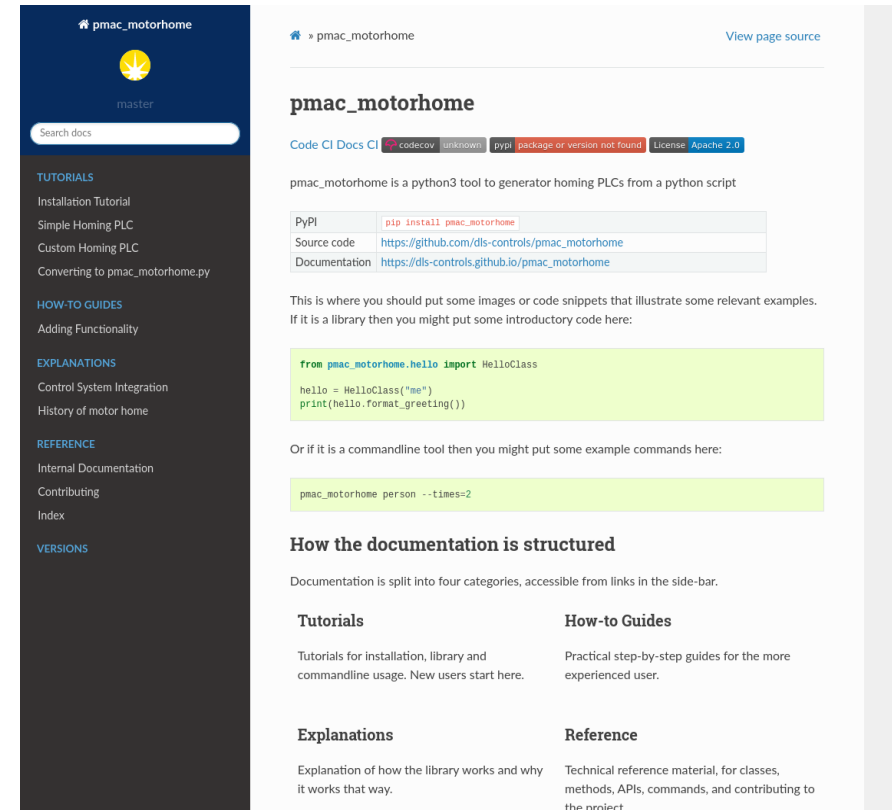
A base template is used for the boilerplate setup and post homing tidying, with loops over all axes to be home

```
def home_slits_hsw(posx, negx, posy, negy):  
    """  
    A special sequence for two pairs of slits in which the vertical and horizontal  
    pairs may collide with each other at the extreme of their homing direction.  
  
    - move all axes to the limit away from their homing direction  
    - home both positive axes using home switch or mark  
    - move the positive axes out of the way  
    - home both negative axes using home switch or mark  
    - move the negative axes out of the way  
  
    Args:  
        posx (int): axis number of the positive horizontal motor  
        negx (int): axis number of the negative horizontal motor  
        posy (int): axis number of the positive vertical motor  
        negy (int): axis number of the negative vertical motor  
    """  
    drive_to_limit(homing_direction=False)  
  
    with only_axes(posx, posy):  
        home_hsw()  
        drive_to_limit(homing_direction=False)  
    with only_axes(negx, negy):  
        home_hsw()  
        drive_to_limit(homing_direction=False)
```

- New homing sequences can be added to the pre-defined set of sequences in `pmac_motorhome`
- New snippets of PLC code can be added
- Command line tools can be defined and made available

Documentation contains:

- Tutorials
- How-to guides
- Explanations for understanding
- Reference material



The screenshot shows the documentation website for `pmac_motorhome`. The left sidebar contains a navigation menu with sections: TUTORIALS (Installation Tutorial, Simple Homing PLC, Custom Homing PLC, Converting to `pmac_motorhome.py`), HOW-TO GUIDES (Adding Functionality), EXPLANATIONS (Control System Integration, History of motor home), REFERENCE (Internal Documentation, Contributing, Index), and VERSIONS. The main content area shows the project name `pmac_motorhome` with a search bar and a list of links: Code, CI, Docs, CI, Codecov, unknown, pypi, package or version not found, and License: Apache 2.0. Below this, a description states: "pmac_motorhome is a python3 tool to generator homing PLCs from a python script". A table provides links for PyPI (pip install pmac_motorhome), Source code (https://github.com/dls-controls/pmac_motorhome), and Documentation (https://dls-controls.github.io/pmac_motorhome). A section titled "This is where you should put some images or code snippets that illustrate some relevant examples. If it is a library then you might put some introductory code here:" shows a code snippet: `from pmac_motorhome.hello import HelloClass
hello = HelloClass("me")
print(hello.format_greeting())`. Another section titled "Or if it is a commandline tool then you might put some example commands here:" shows a command: `pmac_motorhome person --times=2`. The bottom section, "How the documentation is structured", explains that documentation is split into four categories: Tutorials (for installation, library and commandline usage), How-to Guides (practical step-by-step guides), Explanations (how the library works), and Reference (technical reference material).