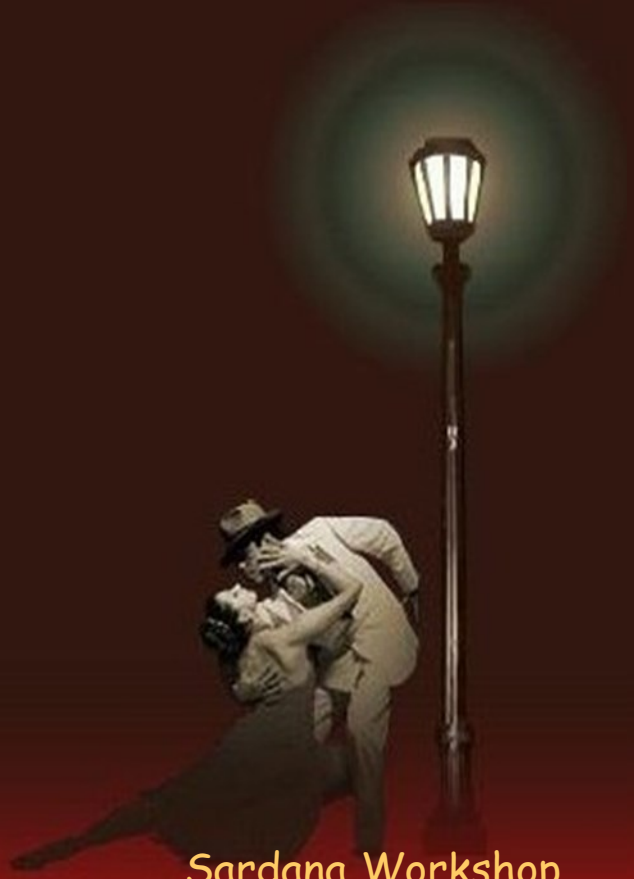
General functions in scan macros

Teresa Núñez
DESY Photon Science

- Motivation
- Implementation:
  - hooks
  - stop function
  - repeating points condition
- Control

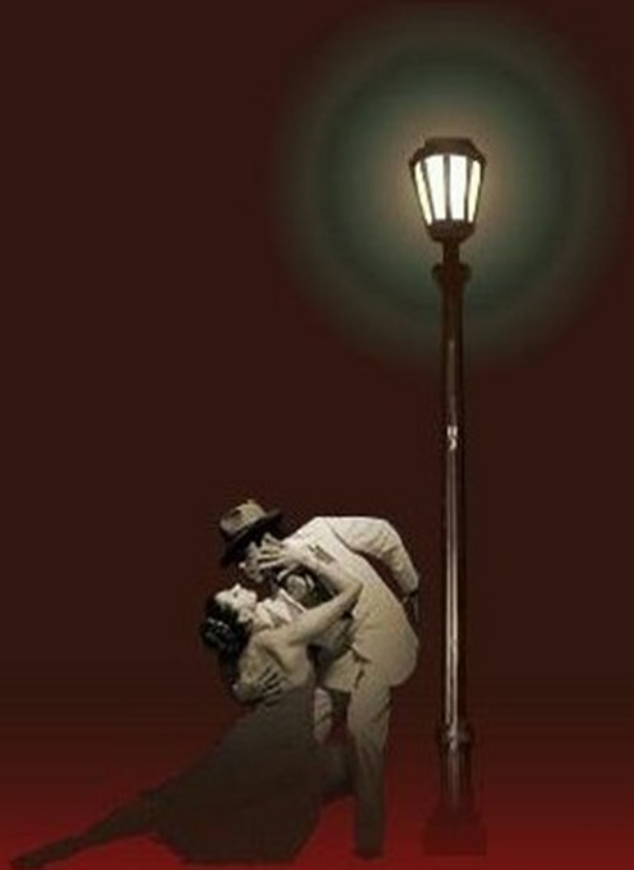


Sardana Workshop
DESY, 06-09-15

Motivation

Customize scans without defining new macros

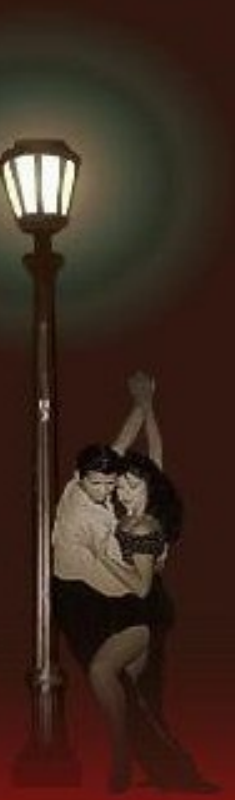
- Applies to all standard scan macros
- Defined by users in a python script



Implementation

- Hooks:
  - Modification of gscan.py
  - Defined in external python file imported by gscan.py
  - Macros select/deselect its use

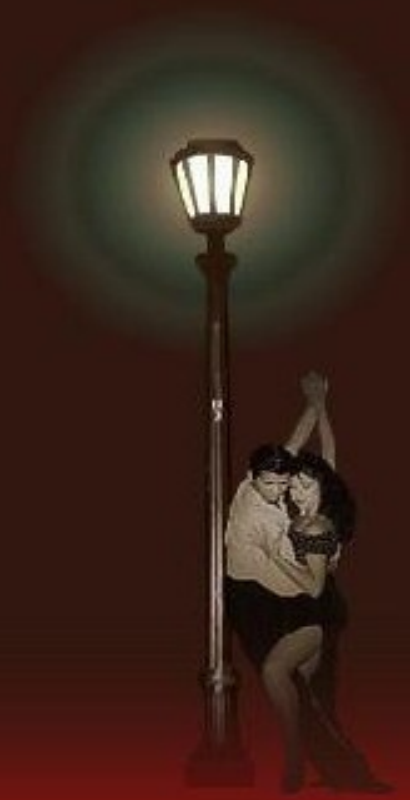
- Stop function (not only for scans):
  - Modification of macro.py and gscan.py
  - Defined in external python file imported by macro.py
  - Macros select/deselect its use



Implementation (ctd.)

- Condition for repeating scan points:
  - Modification of gscan.py
  - Defined in external python file imported by gscan.py
  - Macros select/deselect its use

All points are sent to recorders



```
        elif __builtins__[ 'gh_selector'] == "absorber":
            pmacro.output( "general-pre-scan hook for %s" % __builtins__[ 'gh_selector'])
        else:
            pmacro.output( "general-pre-scan hook, %s" % __builtins__[ 'gh_selector'] )
    else:
        pmacro.output( "general-pre-scan hook, no selector")

def gh_pre_move(pmacro):
    #
    # Example running a macro inside of a general hook
    #
    pmacro.execMacro("mymacro")

    pmacro.output( "Scan pre move")

def gh_post_move(pmacro):
    pmacro.output( "Scan post move")

def gh_pre_acq(pmacro):
    pmacro.output( "Scan pre acq")

def gh_post_acq(pmacro):
    pmacro.output( "Scan post acq")

def gh_post_scan(pmacro):
    pmacro.output( "Scan post scan")


# Function for checking if scan point has to be repeated

def check_condition(pmacro):
    import random
    pmacro.output("Scan check_condition")

    if random.random() > 0.5:
        pmacro.output("Repeat point")
        return 1
    else:
        pmacro.output("DO NOT Repeat point")
        return 0
#
# the general stop function
#
# pmacro.output can not be used because the macro has been stopped
#
def general_on_stop(pmacro):

    if __builtins__.has_key( 'gs_selector'):
        #
        # == 'scan' if we are in a scan, prepared in gscan.py
        #
        if __builtins__[ 'gs_selector'] == "scan":
            print "general_on_stop for scan"
        #
        # == 'general' for all other macros, mv, wa, etc.
        #
        elif __builtins__[ 'gs_selector'] == "general":
            print "general on_stop for general"
        else:
            print "general on_stop"
    else:
        print "General on_stop is called without selector"

#
#  Example without using any selector:
#
#def general_on_stop(pmacro):
#     pmacro.output("General on_stop is called without selector")
#
```

General Functions

General hooks, conditions and on_stop function

It is possible to define certain functions which are executed for each scan. The following features are included: hooks, repeat-a-point-in-condition and on_stop. The code below, `$HOME/sardanaMacros/generalFunctions/general_functions.py` (4.4.1), is a template. This is the Spock interface:

- **gf_status** gives an overview

```
p09/door/haso107d1.01 [1]: gf_status
general features status:
general hooks feature (gh) is disabled
general condition feature (gc) is disabled
general on_stop feature (gs) is disabled

Code: '/home/kracht/sardanaMacros/generalFunctions/general_functions.pyc'

The following macros are involved: gf_status, gf_list, gf_head
All features: gf_enable, gf_disable, gf_set_selector
Individual: XX_enable, XX_disable, XX_isEnabled, XX_setSelector, XX_getSelector
```

Control

Via Macros

- **gf_list** displays `$HOME/sardanaMacros/generalFunctions/general_functions.py`.
- **gf_head** displays the first 20 lines of `$HOME/sardanaMacros/generalFunctions/general_functions.py`.
- **gf_enable, gf_disable, gh_setSelector** control all features:

```
p09/door/haso107d1.01 [2]: %gf_enable
enable general hooks feature
enable general conditions feature
enable general on_stop feature
enable all general features
                Result [2]: True

p09/door/haso107d1.01 [3]: %gf_disable
disable general hooks feature
disable general conditions feature
disable general on_stop feature
disable all general features
                Result [3]: True

p09/door/haso107d1.01 [4]: %gf_setSelector absorber
gh-selector to absorber
gc-setSelector to absorber
on_stop-selector to absorber
                Result [4]: True
```

- **gh_enable, gh_disable, gh_isEnabled, gh_setSelector, gh_getSelector** control the general hooks feature:

```
p09/door/haso107d1.01 [12]: %gh_enable
enable general hooks feature

p09/door/haso107d1.01 [13]: %gh_isEnabled
general hooks feature is enabled
                Result [13]: True

p09/door/haso107d1.01 [14]: %gh_disable
disable general hooks feature

p09/door/haso107d1.01 [15]: %gh_setSelector absorber
gh_setSelector to absorber

p09/door/haso107d1.01 [16]: %gh_getSelector
selector absorber
```

- **gc_enable, gc_disable, gc_isEnabled, gc_setSelector, gc_getSelector** control the general repeat-a-point-on-condition feature. For avoiding repetition of points in the stored data file, the macro **ppDiff** is automatically executed when the scan is finished, leaving only the last data set took for each motor position.

```
p09/door/haso107d1.01 [6]: %gc_enable
enable general conditions feature

p09/door/haso107d1.01 [7]: %gc_disable
disable general conditions feature

p09/door/haso107d1.01 [8]: %gc_isEnabled
general conditions feature is disabled
               Result [8]: False

p09/door/haso107d1.01 [9]: %gc_setSelector align
condition-selector to align

p09/door/haso107d1.01 [10]: %gc_getSelector
condition-selector align
```

- **gs_enable, gs_disable, gs_isEnabled, gs_setSelector, gs_getSelector** control the general on_stop feature.

```
p09/door/haso107d1.01 [11]: %gs_enable
enable general on_stop feature

p09/door/haso107d1.01 [12]: %gs_disable
disable general on_stop feature

p09/door/haso107d1.01 [13]: %gs_isEnabled
general on_stop feature is enabled
               Result [13]: True

p09/door/haso107d1.01 [14]: %gs_setSelector align
on_stop selector to align

p09/door/haso107d1.01 [15]: %gs_getSelector
on_stop selector align
```

  The on_stop feature applies not only to scan but to any macros. Running one scan the gs_selector will be automatically set to 'scan' and reset to 'general' at the end of the scan. This can be used in the user defined on_stop function for having a different behavior when a scan macro or any other macro is aborted.

These macros are part of **handleGeneralFunctions.py** which is distributed at each beamline.

The **selector**s can be used for implementing different behaviors without having the redefine a given function. Depending on to which value(s) the **selector** is set the function will perfome one or another action. There use is completely optional, the code in the examples using them can be simply removed if they don't want to be used.

The selector is a string meant to be used by the functions. One can select only one option, ex.:

```
p09/door/haspp09.01 [1]: gh_setSelector absorber
```

or several ones, ex.:

```
p09/door/haspp09.01 [1]: gh_setSelector absorber_crystal_shutter

p09/door/haspp09.01 [1]: gh_setSelector "absorber crystal shutter"
```

The general functions will look at the value of the selector and perform accordingly.