

# SEP 4 - Motor coupling

Tiago Coutinho

27/08/2009

## Contents

## Introduction

This document describes a sardana enhancement proposal (SEP) for a new “motor coupling“ feature in the Device Pool. The main idea is to be able to trigger multiple motor motions by executing a motion on a single motor.

## Motivation

It would be very useful to be able to trigger multiple motor motions with a single motor motion. Please read the use cases below for a more detailed description.

## Use cases

Consider the case of a 2C Eulerian diffractometer. This diffractometer normally has two physical motors that are usually named theta and 2theta. In most procedures, the diffractometer should operate in a fixed mode where  $2\theta = 2 * \theta$ . This means that the 2theta angle should always be equal to twice the theta angle. Therefore, when moving theta to 20° the system should automatically move 2theta to 40°.

## Pool API

This is a very early proposal of the motor coupling feature. Therefore no concrete API has been proposed yet. What follows is a very low level description of a possible implementation of this feature in the Device Pool.

The motor coupling feature could be implemented by having a “special” motor group that:

- contains all the involved (pseudo) motors
- contains a 'master' (pseudo) motor. All other (pseudo) motor(s) are 'slave(s)'

- it is somehow associated with a calculation method that specifies how to move motors when a master motor is triggered.
- has a enable/disable feature:
  - when enable, if a motion is triggered on the master motor, the motion order is transferred to the motor group which will take the necessary steps to assure that all involved motors are moved to the desired positions. The slave motors are not allowed to move when this motor group is enabled in order to avoid the calculation rule to be broken
  - when disabled, all motors work independently as if this motor group doesn't exist

## Example

Let us consider the use case of the 2C diffractometer given in the above use case.

To implement this use case, a motor group with the following properties should exist:

- two motors th and tth
- th should be the master motor
- a calculation should be implemented like (in python):

```
def calc(new_master_position, current_positions):  
    return { th : new_master_position, tth : 2*new_master_position }
```