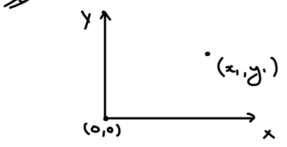
# μCNC or uCNC

For machines with different / custom kinematics, we need to define these 5 functions
/src/hal/kinematics/...

* void kinematics_apply_inverse ( float * $\overset{\text{input}}{\text{axis}}$ , int32_t * $\overset{\text{output}}{\text{steps}}$ );

Converts machine absolute coordinates into step position

Eg: CARTESIAN

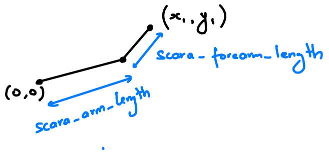


$$\text{axis} = \overset{i_0}{x_1}, \overset{i_1}{y_1} \; \overset{\text{index}}{\ldots}$$

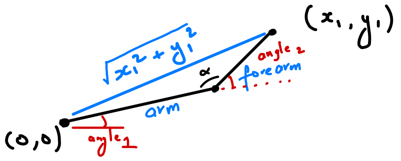
$$\text{g\_settings.steps\_per\_mm}[\,i\,] = x_{spm}, y_{spm}$$

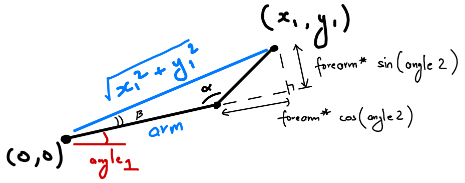
$$\text{steps} = x_1 * x_{spm}, \; y_1 * y_{spm}$$

Eg: SCARA



arm = g_settings.scara_arm_length
forearm = g_settings.scara_forearm_length





Cosine Formula

$$x_1^2 + y_1^2 = arm^2 + forearm^2 - 2.arm.forearm \cos\alpha$$

$$\alpha = \cos^{-1}\left(\frac{x_1^2 + y_1^2 - arm^2 - forearm^2}{-2.arm.forearm}\right)$$

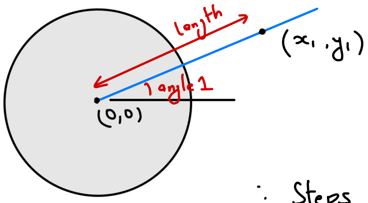
$$\text{Angle2} = \cos^{-1}\left(\frac{x_1^2 + y_1^2 - arm^2 - forearm^2}{2.arm\ forearm}\right)$$

$$\text{Angle 1} = \tan^{-1}\left(\frac{y_1}{x_1}\right) - \underbrace{\tan^{-1}\left(\frac{forearm * \sin(angle2)}{arm + forearm * \cos(angle2)}\right)}_{\beta}$$

$$\therefore \text{Steps} = \text{angle1} * \frac{1}{2\pi} * \text{g\_settings.steps\_per\_mm}[0],$$

$$\text{angle 2} * \frac{1}{2\pi} * \text{g\_settings } \underbrace{\text{steps\_per\_mm}}_{\text{here per revolution}}[1]$$

Eg: MY_PROJECT



$$\text{angle 1} = \tan^{-1}\left(\frac{y_1}{x_1}\right) \qquad \text{Length} = \left|\sqrt{x_1^2 + y_1^2}\right|$$

$$\therefore \text{Steps} = \underbrace{\text{Theta\_ratio}}_{\text{if belt or gear driven}} * \text{angle 1} * \frac{1}{2\pi} * \text{g\_settings.} \underbrace{\text{steps\_per\_mm}}_{\text{here per revolution}}[0] \,,$$

$$\text{length} * \text{g\_settings.steps\_per\_mm}[1]$$

input                                    output

* void kinematics_apply_forward ( int32_t * steps, float * axis )

Converts step position to machine absolute co-ordinates

Eg: CARTESIAN

FORMULA UNIT

$$\text{axis}[i] = \frac{\text{steps}[i]}{\text{steps\_per\_mm}[i]} \qquad \text{mm}$$

Eg: MY_PROJECT

$$\text{angle 1} = \text{steps}[0] * \frac{2\pi}{\underbrace{\text{steps\_per\_mm}[0]}_{\text{here per revolution}}} * \frac{1}{\text{Theta\_ratio}}$$

$$\text{length} = \frac{\text{steps}[1]}{\text{steps\_per\_mm}[1]}$$

$$\therefore \quad x_1, y_1 = \text{Length} * \cos(\text{angle} 1) \,,\; \text{length} * \sin(\text{angle} 1)$$

input

* bool kinematics_check_boundaries ( float * axis ): Checked if inside soft boundaries or not

Return true if soft limits not enabled or if cnc in homing stage
then

Eg. CARTESIAN

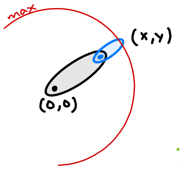
```
for all axis,
        if origin at home position,
                value = axis[i] or - axis[i]  depending on homing direction invert mask
        else
                value = axis[i]

        finally check if all values within range
          i.e. if value > max_distance or value < 0 , return false

return true (within soft boundaries)
```

Eg. SCARA

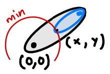
$$\text{distance}^2 = X^2 + Y^2$$



```
if distance² < scara_min_distance² or distance² > scara_max_distance²
    return false
```

Some logic

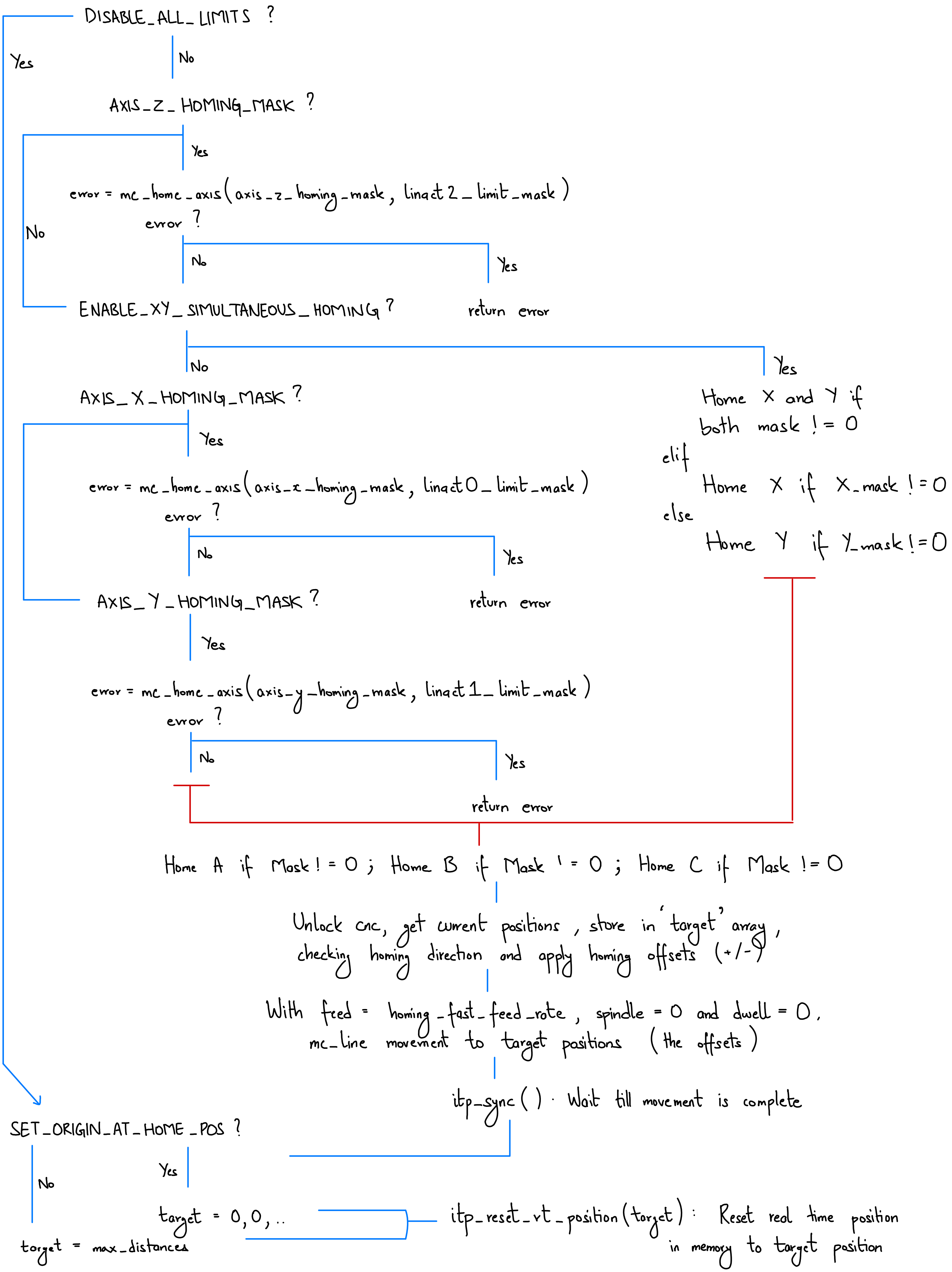
```
for all axis
    ⋮
return true
```



Eg. MY_PROJECT

```
Return true if soft limits not enabled or if cnc in homing stage
                    then
    $distance^2 = x^2 + y^2$
    if $distance^2 > arm\_length^2$
        return false
    { some
      green logic
```

* uint8_t kinematics_home(void) : Homing motion and order of homing of axis.


DISABLE_ALL_LIMITS ?
Yes
No
AXIS_Z_HOMING_MASK ?
Yes
No
error = mc_home_axis(axis_z_homing_mask, linact2_limit_mask)
error ?
No
Yes
ENABLE_XY_SIMULTANEOUS_HOMING ?
return error
No
Yes
Home X and Y if both mask != 0
elif
Home X if X_mask != 0
else
Home Y if Y_mask != 0
AXIS_X_HOMING_MASK ?
Yes
error = mc_home_axis(axis_x_homing_mask, linact0_limit_mask)
error ?
No
Yes
AXIS_Y_HOMING_MASK ?
return error
Yes
error = mc_home_axis(axis_y_homing_mask, linact1_limit_mask)
error ?
No
Yes
return error
Home A if Mask != 0 ; Home B if Mask != 0 ; Home C if Mask != 0
Unlock cnc, get current positions, store in 'target' array, checking homing direction and apply homing offsets (+/-)
With feed = homing_fast_feed_rate, spindle = 0 and dwell = 0, mc_line movement to target positions (the offsets)
itp_sync() : Wait till movement is complete
SET_ORIGIN_AT_HOME_POS ?
No
Yes
target = 0, 0, ..
target = max_distances
itp_reset_rt_position(target) : Reset real time position in memory to target position