2. 机器人运动

2.1. 机器人点动

2.1.1. jog点动

原型	`StartJOG(ref,nb,dir,max_dis,vel=20.0,acc=100.0)`
描述	jog点动
必选参数	• `ref`：0–关节点动,2–基坐标系点动,4–工具坐标系点动,8–工件坐标系点动； • `nb`：1–1关节(x轴),2–2关节(y轴),3–3关节(z轴),4–4关节(rx),5–5关节(ry),6–6关节(rz) • `dir`：0–负方向，1–正方向; • `max_dis`：单次点动最大角度/距离，单位 ° 或 mm;
默认参数	• `vel`：速度百分比，[0~100] 默认20; • `acc`：加速度百分比，[0~100] 默认100;
返回值	错误码 成功–0 失败– errcode

2.1.2. jog点动减速停止

型	`StopJOG(ref)`
述	jog点动减速停止
参数	• `ref`：1–关节点动停止,3–基坐标系点动停止,5–工具坐标系点动停止,9–工件坐标系点动
参数	无
回值	错误码 成功–0 失败– errcode

2.1.3. jog点动立即停止

原型	`ImmStopJOG()`
描述	jog点动立即停止
必选参数	无
默认参数	无
返回值	错误码 成功–0 失败– errcode

2.1.3.1. 代码示例

```python
from fairino import Robot
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
# 机器人单轴点动
robot.StartJOG(0,1,0,20.0,20.0,30.0)    # 单关节运动,StartJOG为非阻塞指令，运动状态下接收其他运动指令（包含StartJOG）会被丢弃
time.sleep(1)
#机器人单轴点动减速停止
ret = robot.StopJOG(1)
print(ret)
#机器人单轴点动立即停止
robot.ImmStopJOG()
robot.StartJOG(0,2,1,20.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(0,3,1,20.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(0,4,1,20.0,vel=40)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(0,5,1,20.0,acc=50)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(0,6,1,20.0,20.0,30.0)
time.sleep(1)
robot.ImmStopJOG()
# 基坐标
robot.StartJOG(2,1,0,20.0)  #基坐标系下点动
time.sleep(1)
# #机器人单轴点动立即停止
robot.ImmStopJOG()
robot.StartJOG(2,1,1,20.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(2,2,1,20.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(2,3,1,20.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(2,4,1,20.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(2,5,1,20.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(2,6,1,20.0)
time.sleep(1)
robot.ImmStopJOG()
# 工具坐标
robot.StartJOG(4,1,0,20.0,20.0,100.0)  #工具坐标系下点动
time.sleep(1)
# #机器人单轴点动立即停止
robot.ImmStopJOG()
robot.StartJOG(4,1,1,20.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,2,1,20.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,3,1,20.0)
```

```
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,4,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,5,1,20.0,vel=10.0,acc=20.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,6,1,20.0,acc=40.0)
time.sleep(1)
robot.ImmStopJOG()
# 工件坐标
robot.StartJOG(8,1,0,20.0,20.0,100.0)  #工件坐标系下点动
time.sleep(1)
# #机器人单轴点动立即停止
robot.ImmStopJOG()
robot.StartJOG(8,1,1,20.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(8,2,1,20.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(8,3,1,20.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(8,4,1,20.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(8,5,1,20.0,vel=30.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(8,6,1,20.0,20.0,acc=90.0)
time.sleep(1)
robot.ImmStopJOG()
```

2.2. 关节空间运动

`MoveJ(joint_pos, tool, user, desc_pos = [0.0,0.0,0.0,0.0,0.0,0.0], vel = 20.0, acc = 0.0, ovl = 100 = [0.0,0.0,0.0,0.0], blendT = -1.0, offset_flag = 0, offset_pos = [0.0,0.0,0.0,0.0,0.0,0.0])`

关节空间运动

- `joint_pos`:目标关节位置，单位[°];
- `tool`:工具号，[0~14];
- `user`:工件号，[0~14];

- `desc_pos`:目标笛卡尔位姿，单位 [mm][°] 默认初值为[0.0,0.0,0.0,0.0,0.0,0.0]，默认值调用正运动
- `vel`:速度百分比，[0~100] 默认20.0;
- `acc`:加速度百分比，[0~100]，暂不开放;
- `ovl`:速度缩放因子，[0~100] 默认100.0;
- `exaxis_pos`:外部轴 1 位置 ~ 外部轴 4 位置 默认[0.0,0.0,0.0,0.0];
- `blendT`:[-1.0]-运动到位 (阻塞)，[0~500.0]-平滑时间 (非阻塞)，单位 [ms] 默认-1.0;
- `offset_flag`:[0]-不偏移，[1]-工件/基坐标系下偏移，[2]-工具坐标系下偏移 默
- `offset_pos`:位姿偏移量，单位 [mm][°] 默认[0.0,0.0,0.0,0.0,0.0,0.0];

错误码 成功-0 失败- errcode

2.2.1. 代码示例

```
from fairino import Robot
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
joint_pos4 = [-83.24, -96.476, 93.688, -114.079, -62, -100]
joint_pos5 = [-43.24, -70.476, 93.688, -114.079, -62, -80]
joint_pos6 = [-83.24, -96.416, 43.188, -74.079, -80, -10]
tool = 0 #工具坐标系编号
user = 0 #工件坐标系编号
ret = robot.MoveJ(joint_pos4, tool, user, vel=30)   #关节空间运动
print("关节空间运动点4:错误码", ret)
ret = robot.MoveJ(joint_pos5, tool, user)
print("关节空间运动点5:错误码", ret)
robot.MoveJ(joint_pos6, tool, user, offset_flag=1, offset_pos=[10,10,10,0,0,0])
print("关节空间运动点6:错误码", ret)
```

2.3. 笛卡尔空间直线运动

`MoveL(desc_pos, tool, user, joint_pos = [0.0,0.0,0.0,0.0,0.0,0.0], vel = 20.0, acc = 0.0 , ovl = blendR = -1.0, exaxis_pos = [0.0,0.0,0.0,0.0], search = 0, offset_flag = 0, offset_pos = [0.0,0.0,0.0,0.0,0.0,0.0],overSpeedStrategy=0,speedPercent=10)`

笛卡尔空间直线运动

- `desc_pos`:目标笛卡尔位姿，单位[mm][°];
- `tool`:工具号，[0~14];
- `user`:工件号，[0~14];

- `joint_pos`:目标关节位置，单位 [°] 默认初值为[0.0,0.0,0.0,0.0,0.0,0.0]，默认值调用逆运动学求
- `vel`:速度百分比，[0~100] 默认20.0;
- `acc`:加速度百分比，[0~100]，暂不开放 默认0.0;
- `ovl`:速度缩放因子，[0~100] 默认100.0;
- `blendR`:blendR:[-1.0]-运动到位 (阻塞)，[0~1000]-平滑半径 (非阻塞)，单位 [mm] 默认-1.0;
- `exaxis_pos`:外部轴 1 位置 ~ 外部轴 4 位置 默认[0.0,0.0,0.0,0.0];
- `search`:[0]-不焊丝寻位，[1]-焊丝寻位;
- `offset_flag`:offset_flag:[0]-不偏移，[1]-工件/基坐标系下偏移，[2]-工具坐标系下偏移 默认（
- `offset_pos`:位姿偏移量，单位 [mm][°] 默认[0.0,0.0,0.0,0.0,0.0,0.0]
- `overSpeedStrategy`:超速处理策略，0-策略关闭；1-标准；2-超速时报错停止；3-自适应降速，
- `speedPercent`:允许降速阈值百分比[0-100]，默认10%

错误码 成功-0 失败- errcode

2.3.1. 代码示例

```python
from fairino import Robot
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
desc_pos1 = [36.794,-475.119, 65.379, -176.938, 2.535, -179.829]
desc_pos2 = [136.794,-475.119, 65.379, -176.938, 2.535, -179.829]
desc_pos3 = [236.794,-475.119, 65.379, -176.938, 2.535, -179.829]
tool = 0 #工具坐标系编号
user = 0 #工件坐标系编号
ret = robot.MoveL(desc_pos1, tool, user)    #笛卡尔空间直线运动
print("笛卡尔空间直线运动点1:错误码", ret)
robot.MoveL(desc_pos2, tool, user, vel=20, acc=100)
print("笛卡尔空间直线运动点2:错误码", ret)
robot.MoveL(desc_pos3, tool, user, offset_flag=1, offset_pos=[10,10,10,0,0,0])
print("笛卡尔空间直线运动点3:错误码", ret)
```

2.4. 笛卡尔空间圆弧运动

```
MoveC(desc_pos_p, tool_p, user_p, desc_pos_t, tool_t, user_t, joint_pos_p =[0.0,0.0,0.0,
0.0,0.0,0.0],joint_pos_t=[0.0,0.0,0.0,0.0,0.0,0.0], vel_p = 20.0,acc_p=100.0, exaxis_pos_p =[0.0,0.
offset_flag_p = 0, offset_pos_p = [0.0,0.0,0.0,0.0,0.0,0.0], vel_t= 20.0, acc_t=100.0,exaxis_pos_t=
[0.0,0.0,0.0,0.0], offset_flag_t = 0, offset_pos_t = [0.0,0.0,0.0, 0.0,0.0,0.0], ovl = 100.0, blend
```

笛卡尔空间圆弧运动

- `desc_pos_p`:路径点笛卡尔位姿，单位[mm][°];
- `tool_p`:路径点工具号，[0~14];
- `user_p`:路径点工件号，[0~14];
- `desc_pos_t`:目标点笛卡尔位姿，单位 [mm][°];
- `tool_t`:工具号，[0~14];
- `user_t`:工件号，[0~14];

- `joint_pos_p`:路径点关节位置，单位 [°] 默认初值为[0.0,0.0,0.0,0.0,0.0,0.0]，默认值调用逆运动学
- `joint_pos_t`:目标点关节位置，单位 [°] 默认初值为[0.0,0.0,0.0,0.0,0.0,0.0]，默认值调用逆运动学
- `vel_p`:路径点速度百分比，[0~100] 默认20.0;
- `acc_p`:路径点加速度百分比，[0~100] 暂不开放,默认0.0;
- `exaxis_pos_p`:路径点外部轴 1位置 ~ 外部轴 4 位置 默认[0.0,0.0,0.0,0.0];
- `offset_flag_p`:路径点是否偏移[0]–不偏移，[1]–工件/基坐标系下偏移，[2]–工具坐标系下偏移 默
- `vel_t`:目标点速度百分比，[0~100] 默认20.0;
- `acc_t`:目标点加速度百分比，[0~100] 暂不开放 默认0.0;
- `exaxis_pos_t`:目标点外部轴 1 位置 ~ 外部轴 4 位置 默认[0.0,0.0,0.0,0.0];
- `offset_flag_t`:目标点是否偏移[0]–不偏移，[1]–工件/基坐标系下偏移，[2]–工具坐标系下偏移 默
- `offset_pos_t`:目标点位姿偏移量，单位 [mm][°] 默认[0.0,0.0,0.0,0.0,0.0,0.0];
- `ovl:`:速度缩放因子，[0~100] 默认100.0;
- `blendR`:[–1.0]–运动到位 (阻塞)，[0~1000]–平滑半径 (非阻塞)，单位 [mm] 默认–1.0;

错误码 成功–0 失败– errcode

2.4.1. 代码示例

```
from fairino import Robot
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
desc_pos1 = [236.794,-475.119, 65.379, -176.938, 2.535, -179.829]
desc_posc1 = [266.794,-455.119, 65.379, -176.938, 2.535, -179.829] #MoveC过渡点
desc_posc2 = [286.794,-475.119, 65.379, -176.938, 2.535, -179.829]  #MoveC目标点
tool = 0#工具坐标系编号
user = 0 #工件坐标系编号
ret = robot.MoveL(desc_pos1, tool, user, vel=30, acc=100)
print("笛卡尔空间直线运动:错误码", ret)
ret = robot.MoveC(desc_posc1, tool, user, desc_posc2,tool, user)  #笛卡尔空间圆弧运动
print("笛卡尔空间圆弧运动:错误码", ret)
```

2.5. 笛卡尔空间整圆运动

```
Circle(desc_pos_p,tool_p,user_p,desc_pos_t,tool_t,user_t,joint_pos_p=[0.0,0.0,0.0,0.0,0.0,0.0], jo
[0.0,0.0,0.0,0.0,0.0,0.0], vel_p = 20.0, acc_p=0.0, exaxis_pos_p= [0.0,0.0, 0.0,0.0], vel_t=20.0, a
exaxis_pos_t =[0.0,0.0,0.0,0.0], ovl=100.0, offset_flag=0, offset_pos= [0.0,0.0,0.0,0.0,0.0,0.0])
```

笛卡尔空间整圆运动

- `desc_pos_p`:路径点笛卡尔位姿，单位[mm][°];
- `tool_p`:工具号，[0~14];
- `user_p`:工件号，[0~14];
- `desc_pos_t`:目标点笛卡尔位姿，单位[mm][°];
- `tool_t`:工具号，[0~14];
- `user_t`:工件号，[0~14];

- `joint_pos_p`:路径点关节位置，单位 [°] 默认初值为[0.0,0.0,0.0,0.0,0.0,0.0]，默认值调用逆运动学
- `joint_pos_t`:目标点关节位置，单位 [°] 默认初值为[0.0,0.0,0.0,0.0,0.0,0.0]，默认值调用逆运动学
- `vel_p`:速度百分比，[0~100] 默认20.0;
- `acc_p`:路径点加速度百分比，[0~100] 暂不开放 默认0.0;
- `exaxis_pos_p`:路径点外部轴 1 位置 ~ 外部轴 4 位置 默认[0.0,0.0,0.0,0.0];
- `vel_t`:目标点速度百分比，[0~100] 默认20.0;
- `acc_t`:目标点加速度百分比，[0~100] 暂不开放 默认0.0;
- `exaxis_pos_t`:标点外部轴 1 位置 ~ 外部轴 4 位置 默认[0.0,0.0,0.0,0.0]
- `ovl`:速度缩放因子，[0~100] 默认100.0;
- `offset_flag`:是否偏移[0]–不偏移，[1]–工件/基坐标系下偏移，[2]–工具坐标系下偏移 默认 0;
- `offset_pos`:位姿偏移量，单位 [mm][°] 默认[0.0,0.0,0.0,0.0,0.0,0.0]

错误码 成功–0 失败– errcode

2.5.1. 代码示例

```
from fairino import Robot
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
desc_pos2 = [236.794,-475.119, 65.379, -176.938, 2.535, -179.829]
desc_posc3 = [256.794,-435.119, 65.379, -176.938, 2.535, -179.829]   #Circle路径点
desc_posc4 = [286.794,-475.119, 65.379, -176.938, 2.535, -179.829]  #Circle目标点
tool = 0#工具坐标系编号
user = 0 #工件坐标系编号
robot.MoveL(desc_pos2, tool, user, vel=40, acc=100)
print("笛卡尔空间直线运动:错误码", ret)
ret = robot.Circle(desc_posc3, tool, user, desc_posc4, tool, user, vel_t=40,
offset_flag=1, offset_pos=[5,10,15,0,0,1])  #笛卡尔空间圆弧运动
print("笛卡尔空间圆弧运动:错误码", ret) #笛卡尔空间整圆运动
```

2.6. 笛卡尔空间螺旋线运动

```
param, joint_pos = [0.0,0.0,0.0,0.0,0.0,0.0], vel = 20.0, acc = 0.0, exaxis_pos = [0.0,0.0,0.0,0.0],
```

位[mm][°];

```
le, rad_init, rad_add, rotaxis_add,
```

螺旋圈数;circle_angle: 螺旋倾角;rad_init: 螺旋初始半径;rad_add: 半径增量;rotaxis_add: 转轴方向增量

立 [°] 默认初值为[0.0,0.0,0.0,0.0,0.0,0.0]，默认值调用逆运动学求解返回值;
认20.0;
默认100.0;
卜部轴 4 位置 默认[0.0,0.0,0.0,0.0];
默认100.0;
-工件/基坐标系下偏移，[2]–工具坐标系下偏移 默认 0;
[mm][°] 默认[0.0,0.0,0.0,0.0,0.0,0.0]

2.6.1. 代码示例

```
from fairino import Robot
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
desc_pos_spiral= [236.794,-475.119, -65.379, -176.938, 2.535, -179.829]#Spiral目标点
#螺旋线参数[circle_num,circle_angle,rad_init,rad_add,rotaxis_add,rot_direction]
# circle_num:螺旋圈数，circle_angle:螺旋倾角，rad_init:螺旋初始半径，rad_add:半径增量，
# rotaxis_add:转轴方向增量，rot_direction:旋转方向，0-顺时针，1-逆时针
param = [5.0,10,30,10,5,0]
tool = 0#工具坐标系编号
user = 0 #工件坐标系编号
ret = robot.NewSpiral(desc_pos_spiral, tool, user, param,vel=40 )  #笛卡尔空间螺旋线运动
print("笛卡尔空间螺旋线运动:错误码", ret)
```

2.7. 伺服运动开始

原型	`ServoMoveStart()`
描述	伺服运动开始，配合ServoJ、ServoCart指令使用
必选参数	无
默认参数	无
返回值	错误码 成功-0 失败- errcode

2.8. 伺服运动结束

原型	`ServoMoveEnd()`
描述	伺服运动结束，配合ServoJ、ServoCart指令使用
必选参数	无
默认参数	无
返回值	错误码 成功-0 失败- errcode

2.9. 关节空间伺服模式运动

原型	`ServoJ(joint_pos, axisPos, acc = 0.0, vel = 0.0, cmdT = 0.008, filterT = 0.0, gain = 0.0)`
描述	关节空间伺服模式运动
必选参数	• `joint_pos`:目标关节位置，单位[°]； • `axisPos`:外部轴位置,单位mm；

默认参数	• `acc`:加速度，范围 [0~100]，暂不开放，默认为 0.0; • `vel`:速度，范围 [0~100]，暂不开放，默认为 0.0; • `cmdT`:指令下发周期，单位s，建议范围[0.001~0.0016], 默认为0.008; • `filterT`:滤波时间，单位 [s]，暂不开放， 默认为0.0; • `gain`:目标位置的比例放大器，暂不开放， 默认为0.0;
返回值	错误码 成功–0 失败– errcode

2.10. 笛卡尔空间伺服模式运动

	`ServoCart(mode, desc_pos, pos_gain = [1.0, 1.0, 1.0, 1.0, 1.0, 1.0] , acc = 0.0, vel = 0.0, c 0.008, filterT = 0.0, gain = 0.0)`
	笛卡尔空间伺服模式运动
数	• `mode`:[0]–绝对运动(基坐标系)，[1]–增量运动(基坐标系)，[2]–增量运动(工具坐标系); • `desc_pos`:目标笛卡尔位置/目标笛卡尔位置增量;
数	• `pos_gain`:位姿增量比例系数，仅在增量运动下生效，范围 [0~1], 默认为 [1.0, 1.0, 1.0, 1.0, 1. • `acc`:加速度，范围 [0~100]，暂不开放，默认为 0.0; • `vel`:速度，范围 [0~100]，暂不开放，默认为 0.0; • `cmdT`:指令下发周期，单位s，建议范围[0.001~0.0016], 默认为0.008; • `filterT`:滤波时间，单位 [s]，暂不开放， 默认为0.0; • `gain`:目标位置的比例放大器，暂不开放， 默认为0.0;
i	错误码 成功–0 失败– errcode

2.10.1. 代码示例

```
from fairino import Robot
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
error,joint_pos = robot.GetActualJointPosDegree()
print("机器人当前关节位置",joint_pos)
joint_pos = [joint_pos[0],joint_pos[1],joint_pos[2],joint_pos[3],joint_pos[4],joint_pos[5]]
error_joint = 0
count =100
error = robot.ServoMoveStart()  #伺服运动开始
print("伺服运动开始错误码",error)
while(count):
    error = robot.ServoJ(joint_pos=joint_pos,axisPos=[0,0,0,0,0,0])   #关节空间伺服模式运动
    if error!=0:
        error_joint =error
    joint_pos[0] = joint_pos[0] + 0.1  #每次1轴运动0.1度，运动100次
    count = count - 1
    time.sleep(0.008)
print("关节空间伺服模式运动错误码",error_joint)
error = robot.ServoMoveEnd()  #伺服运动结束
print("伺服运动结束错误码",error)
mode = 2  #[0]-绝对运动(基坐标系)，[1]-增量运动(基坐标系)，[2]-增量运动(工具坐标系)
n_pos = [0.0,0.0,0.5,0.0,0.0,0.0]   #笛卡尔空间位姿增量
error,desc_pos = robot.GetActualTCPPose()
print("机器人当前笛卡尔位置",desc_pos)
count = 100
error_cart =0
error = robot.ServoMoveStart()  #伺服运动开始
print("伺服运动开始错误码",error)
while(count):
    error = robot.ServoCart(mode, n_pos, vel=40)   #笛卡尔空间伺服模式运动
    if error!=0:
        error_cart =error
    count = count - 1
    time.sleep(0.008)
print("笛卡尔空间伺服模式运动错误码", error_cart)
error = robot.ServoMoveEnd()  #伺服运动开始
print("伺服运动结束错误码",error)
```

2.11. 笛卡尔空间点到点运动

原型	`MoveCart(desc_pos, tool, user, vel = 20.0, acc = 0.0, ovl = 100.0, blendT = -1.0, config = -1)`
描述	笛卡尔空间点到点运动
必选参数	• `desc_pos`:目标笛卡尔位置； • `tool`:工具号，[0~14]； • `user`:工件号，[0~14]；

默认参数	• `vel`:速度，范围 [0~100]，默认为 20.0; • `acc`:加速度，范围 [0~100]，暂不开放,默认为 0.0; • `ovl`:速度缩放因子，[0~100]，默认为 100.0; • `blendT`:[-1.0]-运动到位 (阻塞)，[0~500]-平滑时间 (非阻塞)，单位 [ms] 默认为 -1.0 • `config`:关节配置，[-1]-参考当前关节位置求解，[0~7]-依据关节配置求解 默认为 -1
返回值	错误码 成功-0 失败- errcode

2.11.1. 代码示例

```python
from fairino import Robot
import time
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
desc_pos7 = [236.794,-475.119, 65.379, -176.938, 2.535, -179.829]
desc_pos8 = [236.794,-575.119, 165.379, -176.938, 2.535, -179.829]
desc_pos9 = [236.794,-475.119, 265.379, -176.938, 2.535, -179.829]
tool = 0 #工具坐标系编号
user = 0 #工件坐标系编号
robot.MoveCart(desc_pos7, tool, user)
print("笛卡尔空间点到点运动点7:错误码", ret)
robot.MoveCart(desc_pos8, tool, user, vel=30)
print("笛卡尔空间点到点运动点8:错误码", ret)
robot.MoveCart(desc_pos9, tool, user,)
print("笛卡尔空间点到点运动点9:错误码", ret)
```

2.12. 机器人样条运动

2.12.1. 样条运动开始

原型	`SplineStart()`
描述	样条运动开始
必选参数	无
默认参数	无
返回值	错误码 成功-0 失败- errcode

2.12.2. 样条运动PTP

```
SplinePTP(joint_pos, tool, user, desc_pos = [0.0,0.0,0.0,0.0,0.0,0.0],  vel = 20.0,  acc = 100.0, o
```

样条运动PTP

- `joint_pos`:目标关节位置，单位[°];
- `tool`:工具号，[0~14];
- `user`:工件号，[0~14];

- `desc_pos`:目标笛卡尔位姿，单位 [mm][°] 默认初值为[0.0,0.0,0.0,0.0,0.0,0.0]，默认值调用正运动
- `vel`:速度，范围 [0~100]，默认为 20.0;
- `acc`:加速度，范围 [0~100]，默认为 100.0;
- `ovl`:速度缩放因子，[0~100]，默认为 100.0

错误码 成功–0 失败– errcode

2.12.3. 样条运动结束

原型	`SplineEnd()`
描述	样条运动结束
必选参数	无
默认参数	无
返回值	错误码 成功–0 失败– errcode

2.12.3.1. 代码示例

```python
from fairino import Robot
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
tool = 0 #工具坐标系编号
user = 0 #工件坐标系编号
joint_pos1 = [116.489,-85.278,111.501,-112.486,-85.561,24.693]
joint_pos2 = [86.489,-65.278,101.501,-112.486,-85.561,24.693]
joint_pos3 = [116.489,-45.278,91.501,-82.486,-85.561,24.693]
ret = robot.SplineStart() #样条运动开始
print("样条运动开始:错误码", ret)
ret = robot.SplinePTP(joint_pos1, tool, user)   #样条运动PTP
print("样条运动PTP运动点1:错误码", ret)
ret = robot.SplinePTP(joint_pos2, tool, user)   #样条运动PTP
print("样条运动PTP运动点2:错误码", ret)
ret = robot.SplinePTP(joint_pos3, tool, user)   #样条运动PTP
print("样条运动PTP运动点3:错误码", ret)
ret = robot.SplineEnd() #样条运动结束
print("样条运动结束:错误码", ret)
```

2.13. 机器人新样条运动

2.13.1. 新样条运动开始

在 python 版本发生变更: SDK–v2.0.3

原型	`NewSplineStart(type,averageTime=2000)`
描述	新样条运动开始
必选参数	• `type`:0–圆弧过渡，1–给定点位路径点
默认参数	• `averageTime`:全局平均衔接时间（ms）默认为 2000
返回值	错误码 成功–0 失败– errcode

2.13.2. 新样条运动结束

原型	`NewSplineEnd()`
描述	新样条运动结束
必选参数	无
默认参数	无
返回值	错误码 成功–0 失败– errcode

2.13.3. 新样条指令点

`NewSplinePoint(desc_pos,tool,user,lastFlag,joint_pos=[0.0,0.0,0.0,0.0,0.0,0.0], vel = 0.0, acc = = 100.0 ,blendR = 0.0 )`

新样条指令点

- `desc_pos`:目标笛卡尔位姿，单位 [mm][°];
- `tool`:工具号，[0~14];
- `user`:工件号，[0~14];
- `lastFlag`:是否为最后一个点，0–否，1–是;

- `joint_pos`:目标关节位置，单位 [°] 默认初值为[0.0,0.0,0.0,0.0,0.0,0.0]，默认值调用逆运动学求
- `vel`:速度，范围 [0~100]，暂不开放，默认为 0.0;;
- `acc`:加速度，范围 [0~100]，暂不开放，默认为 0.0;
- `ovl`:速度缩放因子，[0~100] 默认为 100.0;
- `blendR`: [0~1000]–平滑半径，单位 [mm] 默认0.0;

错误码 成功–0 失败– errcode

2.13.3.1. 代码示例

```python
from fairino import Robot
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
tool = 0 #工具坐标系编号
user = 0 #工件坐标系编号
lastFlag= 0 # 是否为最后一个点，0-否，1-是
desc_pos4 = [236.794,-375.119, 65.379, -176.938, 2.535, -179.829]
desc_pos5 = [236.794,-275.119, 165.379, -176.938, 2.535, -179.829]
desc_pos6 = [286.794,-375.119, 265.379, -176.938, 2.535, -179.829]
ret = robot.NewSplineStart(1) #新样条运动开始
print("新样条运动开始:错误码", ret)
ret = robot.NewSplinePoint(desc_pos4, tool, user, lastFlag)#新样条指令点
print("新样条指令点4:错误码", ret)
ret = robot.NewSplinePoint(desc_pos5, tool, user, lastFlag, vel=30)#新样条指令点
print("新样条指令点5:错误码", ret)
lastFlag = 1
ret = robot.NewSplinePoint(desc_pos6, tool, user, lastFlag, vel=30)#新样条指令点
print("新样条指令点6:错误码", ret)
ret = robot.NewSplineEnd() #新样条运动结束
print("新样条运动结束:错误码", ret)
```

2.14. 机器人终止运动

原型	`StopMotion()`
描述	终止运动，使用终止运动需运动指令为非阻塞状态
必选参数	无
默认参数	无
返回值	错误码 成功-0 失败- errcode

2.14.1. 代码示例

```python
from fairino import Robot
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
desc_pos1 = [-187.519, 319.248, 397, -157.278, -31.188, 107.199]
desc_pos2 = [-187.519, 310.248, 297, -157.278, -31.188, 107.199]
joint_pos1 = [-83.24, -96.476, 93.688, -114.079, -62, -100]
tool = 0 #工具坐标系编号
user = 0 #工件坐标系编号
ret = robot.MoveL(desc_pos1, tool, user, joint_pos=joint_pos1)    #笛卡尔空间直线运动
print("笛卡尔空间直线运动点1:错误码", ret)
ret = robot.StopMotion()  #终止运动
print("终止运动:错误码", ret)
robot.MoveL(desc_pos2, tool, user, vel=40, acc=100)
print("笛卡尔空间直线运动点2:错误码", ret)
```

2.15. 机器人暂停运动

latest

在 Python 版本加入: SDK-v2.0.8-3.7.8

原型	`PauseMotion()`
描述	暂停运动，使用暂停运动需运动指令为非阻塞状态
必选参数	无
默认参数	无
返回值	错误码 成功-0 失败- errcode

2.16. 机器人恢复运动

在 Python 版本加入: SDK-v2.0.8-3.7.8

原型	`ResumeMotion()`
描述	恢复运动，使用恢复运动需运动指令为非阻塞状态
必选参数	无
默认参数	无
返回值	错误码 成功-0 失败- errcode

2.17. 机器人点位整体偏移

2.17.1. 点位整体偏移开始

原型	`PointsOffsetEnable(flag,offset_pos)`
描述	点位整体偏移开始
必选参数	• `flag`:0-基坐标或工件坐标系下偏移， 2-工具坐标系下偏移； • `offset_pos`:偏移量，单位[mm][°]。
默认参数	无
返回值	错误码 成功-0 失败- errcode

2.17.2. 点位整体偏移结束

原型	`PointsOffsetDisable()`
描述	点位整体偏移结束
必选参数	无
默认参数	无
返回值	错误码 成功-0 失败- errcode

2.17.2.1. 代码示例

```python
from fairino import Robot
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
desc_pos3 = [-127.519, 256.248, 312, -147.278, -51.588, 107.199]
desc_pos4 = [-140.519, 219.248, 300, -137.278, -11.188, 127.199]
desc_pos5 = [-187.519, 319.248, 397, -157.278, -31.188, 107.199]
desc_pos6 = [-207.519, 229.248, 347, -157.278, -31.188, 107.199]
tool = 0 #工具坐标系编号
user = 0 #工件坐标系编号
flag = 1  #0-基坐标系下/工件坐标系下偏移，2-工具坐标系下偏移
offset_pos = [10,20,30,0,0,0]  #位姿偏移量
ret = robot.PointsOffsetEnable(flag,offset_pos)
print("点位整体偏移开始:错误码", ret)
robot.MoveL(desc_pos3, tool, user, offset_flag=1, offset_pos=[10,10,10,0,0,0])
print("笛卡尔空间直线运动点3:错误码", ret)
robot.MoveL(desc_pos4, tool, user, vel=30, acc=100)
print("笛卡尔空间直线运动点4:错误码", ret)
robot.MoveL(desc_pos5, tool, user)
print("笛卡尔空间直线运动点5:错误码", ret)
ret = robot.PointsOffsetDisable()
print("点位整体偏移结束:错误码", ret)
```

2.18. 控制箱运动AO开始

在 python 版本加入: SDK-v2.0.4

原型	`MoveAOStart(AONum,maxTCPSpeed=1000,maxAOPercent=100,zeroZoneCmp=20)`
描述	控制箱运动AO开始
必选参数	• `AONum`:控制箱AO编号
默认参数	• `maxTCPSpeed`:最大TCP速度值[1-5000mm/s]，默认1000； • `maxAOPercent`:最大TCP速度值对应的AO百分比，默认100%； • `zeroZoneCmp`:死区补偿值AO百分比，整形，默认为20%，范围[0-100]。
返回值	错误码 成功-0 失败- errcode

2.18.1. 代码示例

```
from fairino import Robot
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
#控制箱运动AO开始
error = robot.MoveAOStart(0,100,98,1)
print("MoveAOStart",error)
error,joint_pos = robot.GetActualJointPosDegree()
print("GetActualJointPosDegree",error,joint_pos)
joint_pos[0] = joint_pos[0]+10
#机器人关节运动
error = robot.MoveJ(joint_pos,1,1)
print("MoveJ",error)
time.sleep(3)
#控制箱运动AO停止
error = robot.MoveAOStop()
print("MoveAOStop",error)
```

2.19. 控制箱运动AO结束

在 python 版本加入: SDK–v2.0.4

原型	`MoveAOStop()`
描述	控制箱运动AO结束
必选参数	无
默认参数	无
返回值	错误码 成功–0 失败– errcode

2.20. 末端运动AO开始

在 python 版本加入: SDK–v2.0.4

原型	`MoveToolAOStart(AONum,maxTCPSpeed=1000,maxAOPercent=100,zeroZoneCmp =20)`
描述	末端运动AO开始
必选参数	• `AONum`:末端AO编号
默认参数	• `maxTCPSpeed`:最大TCP速度值[1–5000mm/s]，默认1000； • `maxAOPercent`:最大TCP速度值对应的AO百分比，默认100%； • `zeroZoneCmp`:死区补偿值AO百分比，整形，默认为20%，范围[0–100]。
返回值	错误码 成功–0 失败– errcode

2.20.1. 代码示例

```
from fairino import Robot
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
#末端运动AO开始
error = robot.MoveToolAOStart(0,100,98,1)
print("MoveToolAOStart",error)
error,desc_pos = robot.GetActualTCPPose()
print("GetActualTCPPose",error,desc_pos)
desc_pos[2] = desc_pos[2]-50
#笛卡尔空间直线运动
error = robot.MoveL(desc_pos,1,1)
print("MoveL",error)
time.sleep(3)
#末端运动AO停止
error = robot.MoveToolAOStop()
print("MoveToolAOStop",error)
```

2.21. 末端运动AO结束

在 python 版本加入: SDK–v2.0.4

原型	`MoveToolAOStop()`
描述	末端运动AO结束
必选参数	无
默认参数	无
返回值	错误码 成功–0 失败– errcode

2.22. 开始Ptp运动FIR滤波

在 Python 版本加入: SDK–v2.0.8–3.7.8

原型	`PtpFIRPlanningStart(maxAcc)`
描述	开始Ptp运动FIR滤波
必选参数	• `maxAcc`:最大加速度极值(deg/s2)
默认参数	无
返回值	错误码 成功–0 失败– errcode

2.23. 关闭Ptp运动FIR滤波

在 Python 版本加入: SDK–v2.0.8–3.7.8

原型	`PtpFIRPlanningEnd()`
描述	关闭Ptp运动FIR滤波
必选参数	无
默认参数	无
返回值	错误码 成功-0 失败- errcode

2.23.1. 代码示例

在 Python 版本加入: SDK-v2.0.8-3.7.8

```python
from fairino import Robot
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
startdescPose = [-569.710, -132.595, 395.147, 178.418, -1.893, 171.051]
startjointPos = [-2.334, -79.300, 108.196, -120.594, -91.790, -83.386]
enddescPose = [-366.397, -572.427, 418.339, -178.972, 1.829, -142.970]
endjointPos = [43.651, -70.284, 91.057, -109.075, -88.768, -83.382]
exaxisPos = [0, 0, 0, 0]
offdese = [0, 0, 0, 0, 0, 0]

# Ptp运动FIR滤波开启
robot.PtpFIRPlanningStart(maxAcc=1000)
robot.MoveJ(startjointPos, 0, 0,vel=50)
robot.MoveJ(endjointPos, 0, 0,vel=50)
robot.PtpFIRPlanningEnd()
```

2.24. 开始LIN、ARC运动FIR滤波

在 Python 版本加入: SDK-v2.0.8-3.7.8

原型	`LinArcFIRPlanningStart(maxAccLin,maxAccDeg,maxJerkLin,maxJerkDeg)`
描述	开始LIN、ARC运动FIR滤波
必选参数	• `maxAccLin`:线加速度极值(mm/s2) • `maxAccDeg`:角加速度极值(deg/s2) • `maxJerkLin`:线加加速度极值(mm/s3) • `maxJerkDeg`:角加加速度极值(deg/s3)
默认参数	无
返回值	错误码 成功-0 失败- errcode

2.25. 关闭LIN、ARC运动FIR滤波

在 Python 版本加入: SDK-v2.0.8-3.7.8

原型	`LinArcFIRPlanningEnd()`
描述	关闭LIN、ARC运动FIR滤波
必选参数	无
默认参数	无
返回值	错误码 成功-0 失败- errcode

2.25.1. 代码示例

在 Python 版本加入: SDK-v2.0.8-3.7.8

```python
from fairino import Robot
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')
startdescPose = [-569.710, -132.595, 395.147, 178.418, -1.893, 171.051]
startjointPos = [-2.334, -79.300, 108.196, -120.594, -91.790, -83.386]
enddescPose = [-366.397, -572.427, 418.339, -178.972, 1.829, -142.970]
endjointPos = [43.651, -70.284, 91.057, -109.075, -88.768, -83.382]
exaxisPos = [0, 0, 0, 0]
offdese = [0, 0, 0, 0, 0, 0]

# LIN、ARC运动FIR滤波开启
robot.LinArcFIRPlanningStart(5000, 5000, 5000, 5000)
robot.MoveL(startdescPose, 0, 0,vel=100)
robot.MoveL(enddescPose, 0, 0,vel=100)
robot.LinArcFIRPlanningEnd()
```

2.26. 停止运动

在 python 版本加入: SDK-v2.1.1

原型	`StopMove()`
描述	停止运动
必选参数	无
默认参数	无
返回值	错误码 成功-0 失败- errcode

2.27. 加速度平滑开启

在 python 版本加入: SDK-v2.1.1

原型	`AccSmoothStart(saveFlag_flag)`
描述	加速度平滑开启

必选参数	• `saveFlag_flag`：是否断电保存
默认参数	无
返回值	错误码 成功–0 失败– errcode

2.28. 加速度平滑关闭

在 python 版本加入: SDK–v2.1.1

原型	`AccSmoothEnd(saveFlag_flag)`
描述	加速度平滑关闭
必选参数	• `saveFlag_flag`：是否断电保存
默认参数	无
返回值	错误码 成功–0 失败– errcode

2.28.1. 代码示例

```python
from fairino import Robot
# 与机器人控制器建立连接，连接成功返回一个机器人对象
robot = Robot.RPC('192.168.58.2')

JP1 = [88.927,-85.834,80.289,-85.561,-91.388,108.718]
DP1 = [88.739,-527.617,514.939,-179.039,1.494,70.209]

JP2 = [27.036,-83.909,80.284,-85.579,-90.027,108.604]
DP2 = [-433.125,-334.428,497.139,-179.723,-0.745,8.437]
error = robot.AccSmoothStart(saveFlag=0)
print("AccSmoothStart return:",error)
error = robot.MoveJ(JP1, tool=0, user=0, vel=100)
error = robot.MoveJ(JP2, tool=0, user=0, vel=100)
error = robot.MoveJ(JP1, tool=0, user=0, vel=100)
error = robot.MoveJ(JP2, tool=0, user=0, vel=100)
error = robot.AccSmoothEnd(saveFlag=0)
print("AccSmoothEnd return:", error)
```