

AUBO 机器人 python_sdk 学习资料

(仅供内部学习)

(python2.7 win32)

(AUBOPE V4.3.5)

目录

第一篇 python windows32	5
一、 获取 python SDK 包.....	5
1. 下载 AUBO python windows32 的 sdk 包	5
2. 安装 python	5
二、 运行 python SDK	5
1. 打开 IDLE	5
2. 覆盖 DLL	6
3. 打开 robotcontrol.py	6
4. 运行 robotcontrol.py	7
三、 robotcontrol.py 简要介绍	9
1. 程序结构	9
四、 使用 python win32 SDK	13
五、 python sdk 接口函数示例.....	14
1. get_local_time()	14
2. robot_event_callback(event)	14
3. raise_error(error_type, error_code, error_msg)	14
4. check_event()	15
5. initialize()	15
6. uninitialize()	15
7. Auboi5Robot()	16
8. create_context()作用.....	16
9. get_context()	17
10. connect(ip,port)	17
11. disconnect()	17
12. robot_startup(collision,tool_dynamics)	18
13. robot_shutdown()	18
14. enable_robot_event()	18
15. init_profile()	19
16. set_joint_maxacc(joint_maxacc)	19
17. get_joint_maxacc()	19
18. set_joint_maxvelc(joint_maxvelc)	19
19. get_joint_maxvelc()	20
20. set_end_max_line_acc(end_maxacc)	20
21. get_end_max_line_acc()	20
22. set_end_max_line_velc(end_maxvelc)	20
23. get_end_max_line_velc()	21
24. set_end_max_angle_acc(end_maxacc)	21
25. get_end_max_angle_acc()	21
26. set_end_max_angle_velc(end_maxvelc)	21

27.	get_end_max_angle_velc()	22
28.	move_to_target_in_cartesian(pos, rpy_xyz)	22
29.	move_joint(joint_radian)	22
30.	move_line(joint_radian)	23
31.	move_rotate(user_coord, rotate_axis, rotate_angle)	24
32.	clear_offline_track()	25
33.	append_offline_track_waypoint(waypoints)	25
34.	append_offline_track_file(track_file)	25
35.	startup_offline_track()	25
36.	stop_offline_track()	25
37.	enter_tcp2canbus_mode()	26
38.	leave_tcp2canbus_mode()	26
39.	set_waypoint_to_canbus(joint_radian)	26
40.	remove_all_waypoint()	27
41.	add_waypoint(joint_radian)	27
42.	set_blend_radius(blend_radius)效果不明显	28
43.	set_circular_loop_times(circular_count)	28
44.	set_user_coord(user_coord)用途	28
45.	set_base_coord()用途	28
46.	check_user_coord(user_coord)无效	28
47.	set_relative_offset_on_base(relative_pos, relative_ori)	29
48.	set_relative_offset_on_user(relative_pos, relative_ori, user_coord)	30
49.	set_no_arrival_ahead()	31
50.	set_arrival_ahead_distance(distance)	32
51.	set_arrival_ahead_time(sec)	32
52.	set_arrival_ahead_blend(distance)	33
53.	move_track(track)	33
54.	forward_kin(joint_radian)	36
55.	inverse_kin(joint_radian, pos, ori)	37
56.	base_to_user(pos, ori, user_coord, user_tool)	37
57.	user_to_base(pos, ori, user_coord, user_tool)无效	39
58.	base_to_base_additional_tool(flange_pos, flange_ori, user_tool)	40
59.	rpy_to_quaternion(rpy)	40
60.	quaternion_to_rpy(ori)	41
61.	set_tool_end_param(tool_end_param)用途	41
62.	set_none_tool_dynamics_param()	42
63.	set_tool_dynamics_param(tool_dynamics)	42
64.	get_tool_dynamics_param()单位	42
65.	set_none_tool_kinematics_param()	43
66.	set_tool_kinematics_param(tool_end_param)	43
67.	get_tool_kinematics_param()	43
68.	move_stop()	44

69.	move_pause()	44
70.	move_continue()	44
71.	collision_recover()	44
72.	get_robot_state()	45
73.	set_work_mode(mode)	45
74.	get_work_mode()	45
75.	set_collision_class(grade)	46
76.	is_have_real_robot()	46
77.	is_online_mode()	46
78.	is_online_master_mode()	46
79.	get_joint_status()返回	47
80.	get_current_waypoint()	47
81.	get_board_io_config(io_type)	48
82.	get_board_io_status(io_type, io_name)	48
83.	set_board_io_status(io_type, io_name, io_value)	49
84.	set_tool_power_type(power_type)	49
85.	get_tool_power_type()	49
86.	set_tool_io_type(io_addr, io_type)	50
87.	get_tool_power_voltage()作用	50
88.	get_tool_io_status(io_name)	51
89.	set_tool_io_status(io_name, io_status)	51
90.	startup_excit_traj_track(track_file, track_type, subtype)	51
91.	get_dynidentify_results()	52
92.	set_robot_event_callback(callback)	52
第二篇 python linux32		53
一、	获取 python linux32 的 sdk 包	53
二、	linux 下 python 环境	53
1.	直接终端命令行方式	53
2.	python IDE 方式	53
三、	运行 python SDK	54

第一篇 python windows32

一、 获取 python SDK 包

1. 下载 AUBO python windows32 的 sdk 包

sdk 版本为 python2.7 windows32。

<http://download.aubo-robotics.cn:28080/aubo/download/dev/sdk/v1.2.2/>

Directory Listing For /dev/sdk/v1.2.2/ - Up To /dev/sdk

Filename

[aubo-python-release-1.2.2.zip](#)

[aubo5-sdk-for-windows-x86-csharp-1.2.2.rar](#)





[aubo5-sdk-for-windows-x86-x64-v1.2.2.rar](#)

<http://www.aubo-robotics.cn>

进入目录 `aubo-python-release-1.2.2` -> `win32`。

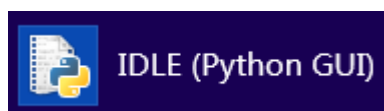
2. 安装 python

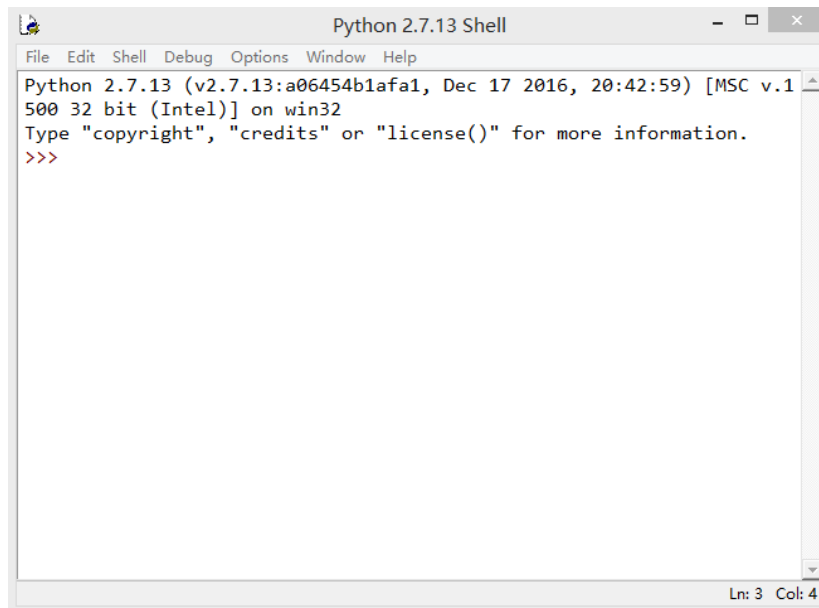
AUBO python SDK 需要在 python2.7.13 上运行。所以先要安装 python。

SDK > python_SDK > aubo-python-release-1.2.2 > win32			▼	🔄
名称	修改日期	类型		
 Dlls	2018/11/28 9:49	文件夹		
 python-2.7.13	2018/2/7 15:33	Windows Insta		
 readme	2018/9/13 14:53	文本文档		
 robotcontrol	2018/8/22 14:54	Python File		

二、 运行 python SDK

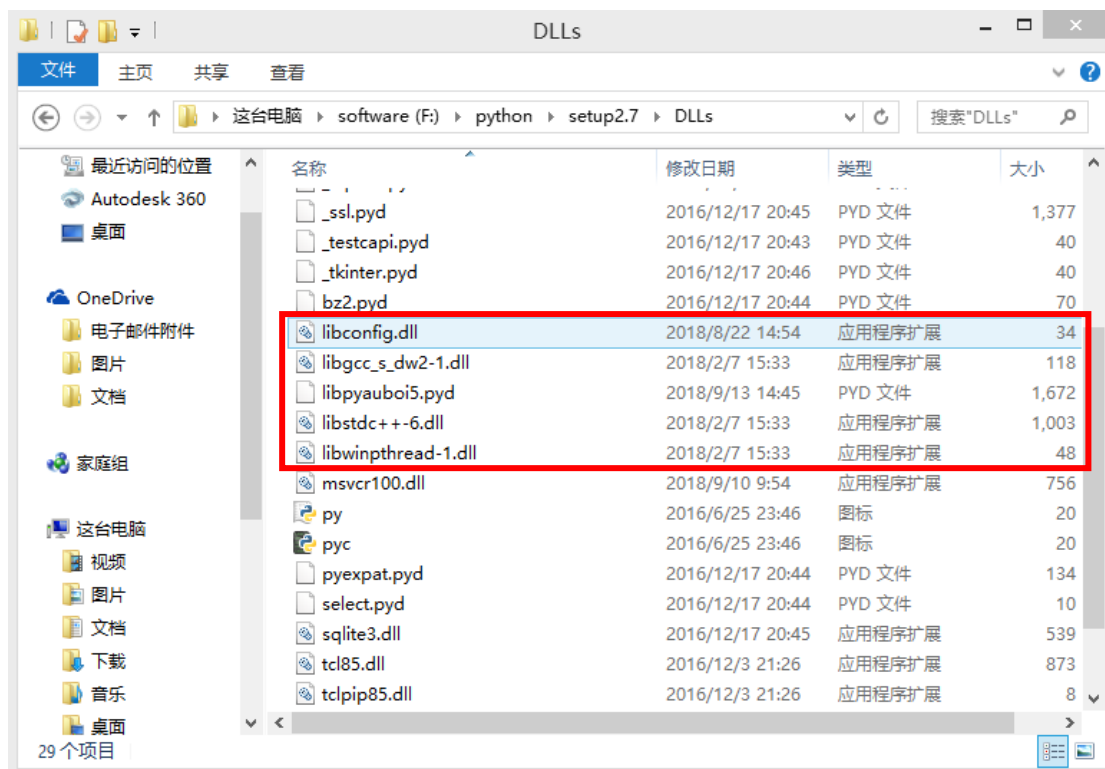
1. 打开 IDLE





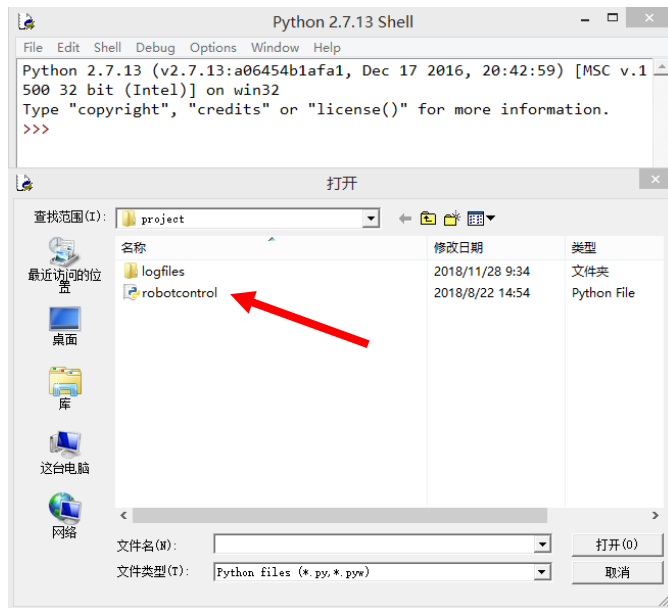
2. 覆盖 DLL

把《win32》目录里的 DLLs 文件夹拷贝到 python 安装目录，覆盖。



3. 打开 robotcontrol.py

robotcontrol.py 是 AUBO python SDK 的测试程序。



```
#!/usr/bin/env python
# coding=utf-8
import time
import libpyauboi5
import logging
from logging.handlers import RotatingFileHandler
import os
from math import pi

# 创建一个logger
logger = logging.getLogger()

def logger_init():
    # Log 等级总开关
    logger.setLevel(logging.INFO)

    # 创建log目录
    if not os.path.exists('./logfiles'):
        os.mkdir('./logfiles')

    # 创建一个handler，用于写入日志文件
    logfile = './logfiles/robot-ctl-python.log'

    # 以append模式打开日志文件
    fh = logging.FileHandler(logfile, mode='a')
    fh = RotatingFileHandler(logfile, mode='a', maxBytes=1024*1024*50, backupCou

    # 输出到file的log等级的开关
    fh.setLevel(logging.INFO)

    # 再创建一个handler，用于输出到控制台
    ch = logging.StreamHandler()

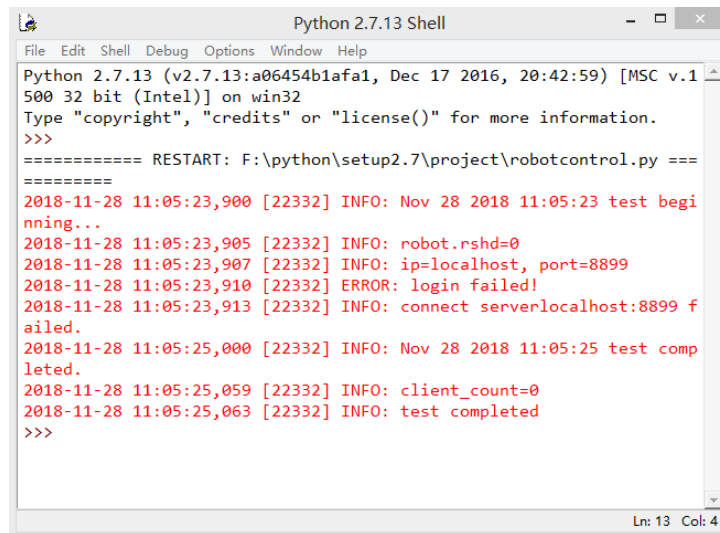
    # 输出到console的log等级的开关
    ch.setLevel(logging.INFO)

    # 定义handler的输出格式
    # formatter = logging.Formatter("%(asctime)s - %(filename)s[line:%(lineno)d]
    formatter = logging.Formatter("%(asctime)s [%(thread)u] %(levelname)s: %(mes

Ln: 1 Col: 0
```

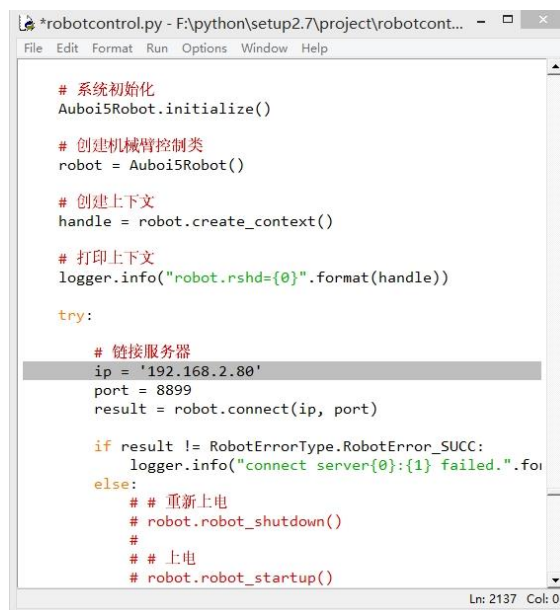
4. 运行 robotcontrol.py

点击【Run】->【Run Module】,在 Python 2.7.13 Shell 里打印出以下信息（由于没有连接到机器人服务器，所以 login failed）:



```
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:\python\setup2.7\project\robotcontrol.py =====
2018-11-28 11:05:23,900 [22332] INFO: Nov 28 2018 11:05:23 test beginning...
2018-11-28 11:05:23,905 [22332] INFO: robot.rshd=0
2018-11-28 11:05:23,907 [22332] INFO: ip=localhost, port=8899
2018-11-28 11:05:23,910 [22332] ERROR: login failed!
2018-11-28 11:05:23,913 [22332] INFO: connect serverlocalhost:8899 failed.
2018-11-28 11:05:25,000 [22332] INFO: Nov 28 2018 11:05:25 test completed.
2018-11-28 11:05:25,059 [22332] INFO: client_count=0
2018-11-28 11:05:25,063 [22332] INFO: test completed
>>>
```

- 连接 PC 到机器人，保证 ping 通。假设机器人 Ip 为 192.168.2.80。
- 修改 robotcontrol.py 的 2137 行程序，ip 改为机器人 ip，保存。
- 点击【Run】->【Run Module】，可以看到机器人在动作，同时 Python 2.7.13 Shell 会打印出许多信息。



```
*robotcontrol.py - F:\python\setup2.7\project\robotcont...
File Edit Format Run Options Window Help

# 系统初始化
Auboi5Robot.initialize()

# 创建机械臂控制类
robot = Auboi5Robot()

# 创建上下文
handle = robot.create_context()

# 打印上下文
logger.info("robot.rshd={0}".format(handle))

try:

# 链接服务器
ip = '192.168.2.80'
port = 8899
result = robot.connect(ip, port)

if result != RobotErrorType.RobotError_SUCC:
    logger.info("connect server{0}:{1} failed.".foi
else:
    ## 重新上电
    # robot.robot_shutdown()
    ## 上电
    # robot.robot_startup()
```


三、 robotcontrol.py 简要介绍

1. 程序结构

大致分为 6 部分：

- 导入模块（line3 ~ line8）。
- 定义日志输出记录函数（line14 ~ line52）。
- 定义枚举类型（line14 ~ line52）。

枚举类型	说明	行号
RobotEventType	机器人事件类型	Ln 55
RobotErrorType	机器人调用返回错误类型	Ln 149
RobotEvent	机器人事件	Ln 164
RobotError	机器人错误	Ln 172
RobotDefaultParameters	机器人缺省参数	Ln 182
RobotMoveTrackType	轨迹运动类型	Ln 197
RobotIOType	IO 类型	Ln 216
RobotToolIoName	工具 IO 名称	Ln 232
RobotUserIoName	用户 IO 名称	Ln 242
RobotStatus	机器人运行状态	Ln 283
RobotRunningMode	机器人运行模式	Ln 297
RobotToolPowerType	工具 IO 电压	Ln 307
RobotToolIoAddr	工具 IO 地址	Ln 316
RobotCoordType	坐标系类型	Ln 326
RobotCoordCalMethod	用户坐标系标定方法	Ln 338
RobotToolDigitalIoDir	工具 IO 方向（input/output）	Ln 353

- Auboi5Robot 类

成员函数	说明	行号
get_local_time	获取系统当前时间	Ln 383
robot_event_callback	机械臂事件	Ln 394
raise_error	抛出异常事件	Ln 409
check_event	检查机械臂是否发生异常事件	Ln 420
initialize	初始化机械臂控制库	Ln 435
uninitialize	反初始化机械臂控制库	Ln 452
create_context	创建机械臂控制上下文句柄	Ln 464
get_context	获取机械臂当前控制上下文	Ln 476
connect	链接机械臂服务器	Ln 487
disconnect	断开机械臂服务器链接	Ln 516
robot_startup	启动机械臂	Ln 535
robot_shutdown	关闭机械臂	Ln 561
enable_robot_event		Ln 579
init_profile	初始化机械臂控制全局属性	Ln 588

set_joint_maxacc	设置六个关节的最大加速度	Ln 606
get_joint_maxacc	获取六个关节的最大加速度	Ln 623
set_joint_maxvelc	设置六个关节的最大速度	Ln 640
get_joint_maxvelc	获取六个关节的最大速度	Ln 657
set_end_max_line_acc	设置机械臂末端最大线加速度	Ln 674
get_end_max_line_acc	获取机械臂末端最大线加速度	Ln 691
set_end_max_line_velc	设置机械臂末端最大线速度	Ln 708
get_end_max_line_velc	获取机械臂末端最大线速度	Ln 725
set_end_max_angle_acc	设置机械臂末端最大角加速度	Ln 742
get_end_max_angle_acc	获取机械臂末端最大角加速度	Ln 759
set_end_max_angle_velc	设置机械臂末端最大角速度	Ln 776
get_end_max_angle_velc	获取机械臂末端最大角速度	Ln 793
move_to_target_in_cartesian	给出笛卡尔坐标值和欧拉角，机械臂轴动到目标位置和姿态	Ln 810
move_joint	机械臂轴动	Ln 852
move_line	机械臂直线运动	Ln 873
move_rotate	保持当前位置变换姿态做旋转运动	Ln 894
clear_offline_track	清理服务器上的非在线轨迹运动数据	Ln 923
append_offline_track_waypoint		Ln 623
append_offline_track_file		Ln 623
startup_offline_track		Ln 623
stop_offline_track		
enter_tcp2canbus_mode		
leave_tcp2canbus_mode		
set_waypoint_to_canbus		
remove_all_waypoint	清除所有已经设置的全局路点	Ln 1061
add_waypoint	添加全局路点用于轨迹运动	Ln 1078
set_blend_radius	设置交融半径	Ln 1095
set_circular_loop_times	设置圆运动圈数	Ln 1116
set_user_coord	设置用户坐标系	Ln 1135
set_base_coord	设置基座坐标系	Ln 1162
check_user_coord	检查用户坐标系参数设置是否合理	Ln 1179
set_relative_offset_on_base	设置基于基座坐标系运动偏移量	Ln 1201
set_relative_offset_on_user	设置基于用户标系运动偏移量	Ln 1219
set_no_arrival_ahead	取消提前到位设置	Ln 1248
set_arrival_ahead_distance	设置距离模式下的提前到位距离	Ln 1270
set_arrival_ahead_time	设置时间模式下的提前到位时间	Ln 1292
set_arrival_ahead_blend	设置距离模式下交融半径距离	Ln 1314
move_track	轨迹运动	Ln 1336
forward_kin	正解	Ln 1360

inverse_kin	逆解	Ln 1379
base_to_user	用户坐标系转基座坐标系	Ln 1401
user_to_base	用户坐标系转基座坐标系	Ln 1428
base_to_base_additional_tool	基坐标系转基座标得到工具末端点的位置和姿态	Ln 1455
rpy_to_quaternion	欧拉角转四元数	Ln 1471
quaternion_to_rpy	四元数转欧拉角	Ln 1488
set_tool_end_param	设置末端工具参数	Ln 1505
set_none_tool_dynamics_param	设置无工具的动力学参数	Ln 1523
set_tool_dynamics_param	设置工具的动力学参数	Ln 1541
get_tool_dynamics_param	获取末端工具参数	Ln 1562
set_none_tool_kinematics_param	设置无工具运动学参数	Ln 1584
set_tool_kinematics_param	设置工具的运动学参数	Ln 1602
get_tool_kinematics_param	设置工具的运动学参数	Ln 1620
move_stop	停止机械臂运动	Ln 1639
move_pause	暂停机械臂运动	Ln 1656
move_continue	暂停后回复机械臂运动	Ln 1673
collision_recover	机械臂碰撞后恢复	Ln 1690
get_robot_state	获取机械臂当前状态	Ln 1707
set_work_mode	设置机械臂服务器工作模式	Ln 1729
get_work_mode	获取机械臂服务器当前工作模式	Ln 1748
set_collision_class	设置机械臂碰撞等级	Ln 1770
is_have_real_robot	获取当前是否已经链接真实机械臂	Ln 1787
is_online_mode	当前机械臂是否运行在联机模式	Ln 1804
is_online_master_mode	当前机械臂是否运行在联机主模式	Ln 1821
get_joint_status	获取机械臂当前状态信息	Ln 1838
get_current_waypoint	获取机械臂当前位置信息	Ln 1862
get_board_io_config	返回 IO 配置	Ln 1882
get_board_io_status	获取 IO 状态	Ln 1905
set_board_io_status	设置 IO 状态	Ln 1923
set_tool_power_type	设置工具端电源类型	Ln 1942
get_tool_power_type	获取工具端电源类型	Ln 1962
set_tool_io_type	设置工具端数字 IO 类型	Ln 1984
get_tool_power_voltage	获取工具端电压数值	Ln 2007
get_tool_io_status	获取工具端 IO 状态	Ln 2024
set_tool_io_status	设置工具端 IO 状态	Ln 2043
startup_excit_traj_track		Ln 1962
get_dynidentify_results		Ln 1962
set_robot_event_callback	设置机械臂事件回调函数	Ln 2096

● 测试程序

测试程序	行号
def test(test_count)	Ln 2114
def step_test()	Ln 2281
def excit_traj_track_test()	Ln 2361
def move_rotate_test()	Ln 2432

- 主函数
选择哪个测试程序。此处选择 **test()** 函数。

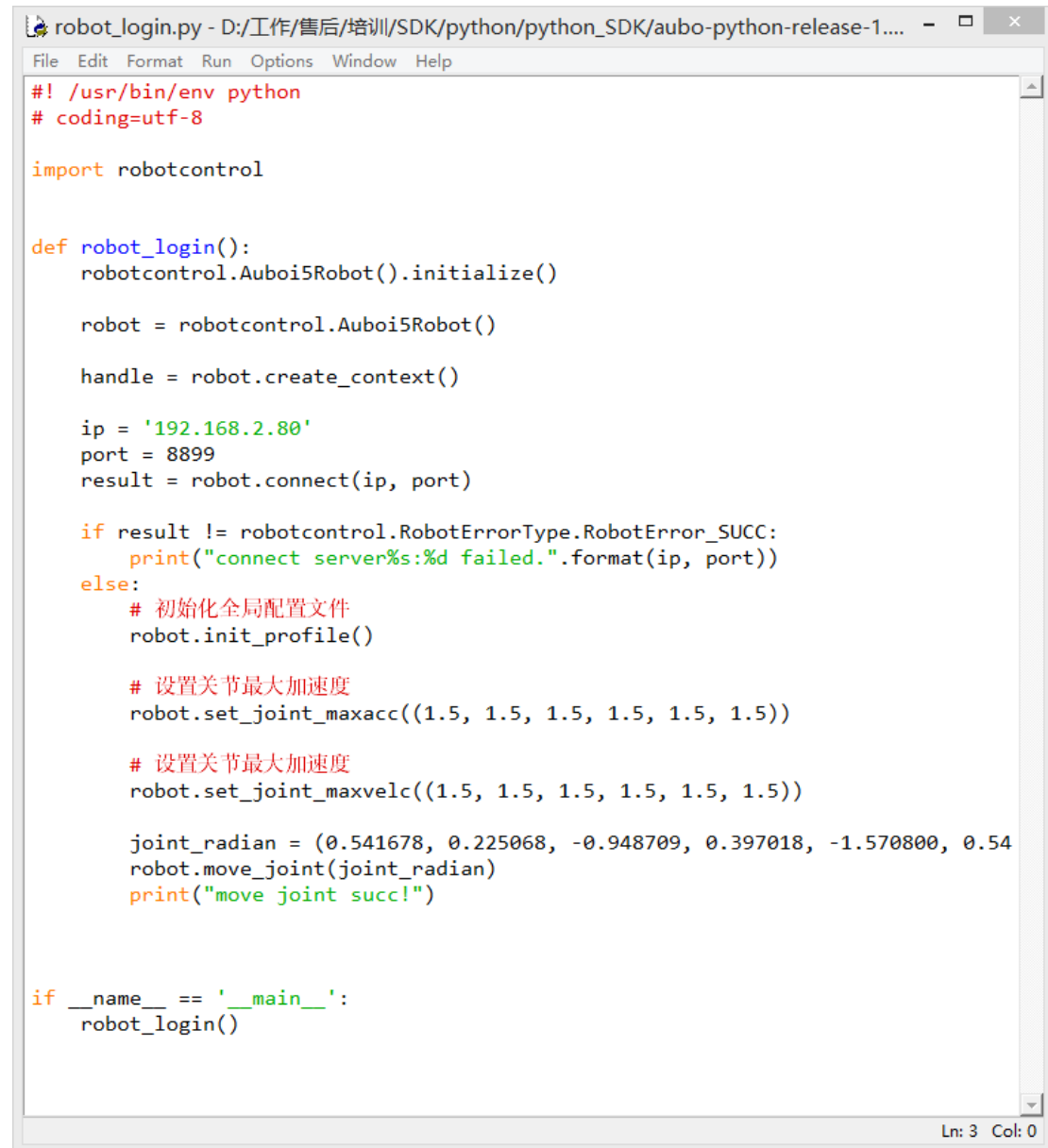
四、 使用 python win32 SDK

利用 python windows32 SDK 编写自己的 python 控制程序。

要使用 python 库，先导入 robotcontrol:

```
import robotcontrol
```

可按照如下程序模板编写:

A screenshot of a Python IDE window titled 'robot_login.py - D:/工作/售后/培训/SDK/python/python_SDK/aubo-python-release-1....'. The window contains a Python script for robot login. The script starts with a shebang line and encoding declaration, followed by importing the robotcontrol module. A function 'robot_login()' is defined, which initializes a robot, connects to a server at 192.168.2.80 on port 8899, and sets joint limits. It then moves the joints to specific radian values and prints a success message. The script is executed as the main function.

```
#!/usr/bin/env python
# coding=utf-8

import robotcontrol

def robot_login():
    robotcontrol.Auboi5Robot().initialize()

    robot = robotcontrol.Auboi5Robot()

    handle = robot.create_context()

    ip = '192.168.2.80'
    port = 8899
    result = robot.connect(ip, port)

    if result != robotcontrol.RobotErrorType.RobotError_SUCC:
        print("connect server%s:%d failed.".format(ip, port))
    else:
        # 初始化全局配置文件
        robot.init_profile()

        # 设置关节最大加速度
        robot.set_joint_maxacc((1.5, 1.5, 1.5, 1.5, 1.5, 1.5))

        # 设置关节最大速度
        robot.set_joint_maxvelc((1.5, 1.5, 1.5, 1.5, 1.5, 1.5))

        joint_radian = (0.541678, 0.225068, -0.948709, 0.397018, -1.570800, 0.54)
        robot.move_joint(joint_radian)
        print("move joint succ!")

if __name__ == '__main__':
    robot_login()
```

Ln: 3 Col: 0

五、 python sdk 接口函数示例

1. get_local_time()

get_local_time()	
说明	获取系统当前时间 格式为字符串
示例	<pre>11 #获取系统时间 12 print(robotcontrol.Auboi5Robot.get_local_time())</pre>
输出	<pre>33 Dec 03 2018 18:39:10</pre>

2. robot_event_callback(event)

robot_event_callback()	
说明	机械臂回调函数
示例	<p>改写 robotcontrol.py 中的程序，如下，使打印出机械臂事件信息</p> <pre>def robot_event_callback(self, event): """ * FUNCTION: robot_event_callback * DESCRIPTION: 机械臂事件 * INPUTS: 无输入 * OUTPUTS: 系统事件回调函数 * RETURNS: 系统事件回调函数 * NOTES: """ if event['type'] not in RobotEventType.NoError: self.last_error = RobotError(event['type'], event['code'], event['content']) else: self.last_event = RobotEvent(event['type'], event['code'], event['content']) print(event)</pre> <p>主程序里添加：</p> <pre>806 #使能事件回调 807 ret = robot.enable_robot_event() 808 print("enable_robot_event ret is {0}".format(ret))</pre>
输出	<pre>1020 {'content': '{"code":0,"text":"RobotMoveControlStopDone."}', 'code': 0, 'type': 4 1021 4} 1022 {'content': '{"code":0,"text":"RobotMoveControlStopDone."}', 'code': 0, 'type': 4 1023 4} 1024 {'content': '{"code":0,"text":"RobotMoveControlStopDone."}', 'code': 0, 'type': 4 1025 4} 1026 {'content': '{"code":0,"text":"RobotMoveControlStopDone."}', 'code': 0, 'type': 4 1027 4} 1028 {'content': '{"code":0,"text":"RobotMoveControlStopDone."}', 'code': 0, 'type': 4 1029 4}</pre>

3. raise_error(error_type, error_code, error_msg)

raise_error(error_type, error_code, error_msg)	
说明	抛出异常事件 内部使用
示例	

输出	
----	--

4. check_event()

check_event()	
说明	内部使用
示例	
输出	

5. initialize()

initialize()	
说明	初始化机械臂控制库 成功返回 0，失败返回其他
示例	<pre> File Edit Format Run Options Window Help #!/usr/bin/env python # coding=utf-8 import robotcontrol def robot_login(): #初始化机械臂控制库 ret = robotcontrol.Auboi5Robot().initialize() print("Auboi5Robot().initialize() is {}".format(ret)) </pre>
输出	Auboi5Robot().initialize() is 0

6. uninitialized()

uninitialize()	
说明	反初始化机械臂控制库 成功返回 0，失败返回其他

示例	<pre> File Edit Format Run Options Window Help #!/usr/bin/env python # coding=utf-8 import robotcontrol def robot_login(): #初始化机械臂控制库 ret = robotcontrol.Auboi5Robot().initialize() print("Auboi5Robot().initialize() is {}".format(ret)) #其他逻辑 #反初始化机械臂控制库 ret = robotcontrol.Auboi5Robot.uninitialize() print("Auboi5Robot().uninitialize() is {}".format(ret)) </pre>
输出	<pre> Auboi5Robot().initialize() is 0 Auboi5Robot().uninitialize() is 0 </pre>

7. Auboi5Robot()

Auboi5Robot()	
说明	实例化一个机械臂控制类
示例	<pre>robot = robotcontrol.Auboi5Robot()</pre>
输出	

8. create_context()作用

create_context()	
说明	创建机械臂控制上下文句柄 成功返回: RSHD 默认的句柄为-1, 每新建一个句柄就+1
示例	<pre> #创建一个句柄 handle_1 = robot.create_context() print("Auboi5Robot().create_context() is {}".format(handle_1)) #创建第二个句柄 handle_2 = robot.create_context() print("Auboi5Robot().create_context() is {}".format(handle_2)) </pre>
输出	<pre> Auboi5Robot().create_context() is 0 Auboi5Robot().create_context() is 1 </pre>

9. get_context()

get_context()	
说明	获取机械臂当前控制上下文
示例	<pre>#实例化一个控制类对象 robot = robotcontrol.Auboi5Robot() #获取当前控制上下文 print("current context is {0}".format(robot.get_context())) #创建一个句柄 handle_1 = robot.create_context() ## print("Auboi5Robot().create_context() is {0}".format(handle_1)) #获取当前控制上下文 print("current context is {0}".format(robot.get_context())) #创建第二个句柄 handle_2 = robot.create_context() ## print("Auboi5Robot().create_context() is {0}".format(handle_2)) #获取当前控制上下文 print("current context is {0}".format(robot.get_context()))</pre>
输出	<pre>current context is -1 current context is 0 current context is 1</pre>

10. connect(ip,port)

connect(ip,port)	
说明	连接登录机器人 ip:机器人 ip port:固定为 8899 远程监控机器人之前必须连接登录
示例	<pre>25 #登录机器人 26 ret = robot.connect("192.168.2.80",8899) 27 print("login ret is {0}".format(ret))</pre>
输出	<pre>50 login ret is 0</pre>

11. disconnect()

disconnect()	
说明	退出登录机器人 完成全部操作（包括机器人断电）后，退出登录
示例	<pre>31 #断开连接 32 ret = robot.disconnect() 33 print("logout ret is {0}".format(ret))</pre>
输出	<pre>58 logout ret is 0</pre>

12. robot_startup(collision,tool_dynamics)

robot_startup(collision=RobotDefaultParameters.collision_grade, tool_dynamics=RobotDefaultParameters.tool_dynamics)		
说明	机械臂上电 collision: 碰撞等级范围(0~10) 缺省: 6 tool_dynamics:工具动力学参数 tool_dynamics = 位置, 单位(m) : {"position": (0.0, 0.0, 0.0), 负载, 单位(kg): "payload": 1.0, 惯量: "inertia": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0)} 成功返回 0	
示例	<pre>38 #机械臂上电,碰撞等级 4, 工具动力学参数(0,0.1,0.05),1.0kg 39 collision = 4 40 tool_dynamics = {"position":(0,0.1,0.05),"payload":1.0,"inertia":(0,0,0,0,0,0)} 41 ret = robot.robot_startup(collision,tool_dynamics) 42 print("robot_startup ret is {0}".format(ret))</pre>	
输出	75	robot_startup ret is 0

13. robot_shutdown()

robot_shutdown()		
说明	机械臂断电 应当保证机械臂完成工作且停稳后再断电 成功返回 0	
示例	<pre>57 #机械臂断电 58 ret = robot.robot_shutdown() 59 print("robot shutdown ret is {0}".format(ret))</pre>	
输出	79	robot shutdown ret is 0

14. enable_robot_event()

enable_robot_event()		
说明	开启机械臂事件回调	
示例	<pre>806 #使能事件回调 807 ret = robot.enable_robot_event() 808 print("enable_robot_event ret is {0}".format(ret))</pre>	
输出	<pre>1020 {'content': '{"code":0,"text":"RobotMoveControlStopDone."}', 'code': 0, 'type': 4 1021 4} 1022 {'content': '{"code":0,"text":"RobotMoveControlStopDone."}', 'code': 0, 'type': 4 1023 4} 1024 {'content': '{"code":0,"text":"RobotMoveControlStopDone."}', 'code': 0, 'type': 4 1025 4} 1026 {'content': '{"code":0,"text":"RobotMoveControlStopDone."}', 'code': 0, 'type': 4 1027 4} 1028 {'content': '{"code":0,"text":"RobotMoveControlStopDone."}', 'code': 0, 'type': 4 1029 4}</pre>	

15. init_profile()

init_profile()		
说明	初始化机械臂运动控制属性，初始化后： joint_maxacc:25°/s ² joint_maxvelc:25°/s end_line_maxacc:0.03m/s end_line_maxvelc:0.03m/s end_angle_maxacc:100°/s ² end_angle_maxvelc:1.7°/s 坐标系:base	
示例	<pre>56 #初始化机器人全局运动属性 57 ret = robot.init_profile() 58 print("robot init profile ret is {}".format(ret))</pre>	
输出	126 robot init profile ret is 0	

16. set_joint_maxacc(joint_maxacc)

set_joint_maxacc(joint_maxacc=(1.0, 1.0, 1.0, 1.0, 1.0, 1.0))		
说明	设置关节运动最大加速度	
示例	<pre>66 #设置最大加速度 67 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 68 ret = robot.set_joint_maxacc(joint_acc) 69 print("robot set_joint_maxacc ret is {}".format(ret))</pre>	
输出	203 robot set_joint_maxacc ret is 0	

17. get_joint_maxacc()

get_joint_maxacc()		
说明	获取机器人关节运动最大加速度，单位 rad	
示例	<pre>66 #设置关节最大加速度 67 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 68 ret = robot.set_joint_maxacc(joint_acc) 69 print("robot set_joint_maxacc ret is {}".format(ret)) 70 #获取关节最大加速度 71 print("joint maxacc:{}".format(robot.get_joint_maxacc()))</pre>	
输出	239 robot set_joint_maxacc ret is 0 240 joint maxacc:[2.0, 2.0, 2.0, 2.0, 2.0, 2.0]	

18. set_joint_maxvelc(joint_maxvelc)

set_joint_maxvelc(joint_maxvelc=(1.0, 1.0, 1.0, 1.0, 1.0, 1.0))		
说明	设置机器人关节运动最大速度	

示例	73	#设置关节最大速度
	74	joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0)
	75	ret = robot.set_joint_maxvelc(joint_velc)
	76	print("robot set_joint_maxvelc ret is {0}".format(ret))
输出	241	robot set_joint_maxvelc ret is 0

19. get_joint_maxvelc()

get_joint_maxvelc()		
说明	获取关节运动最大速度	
示例	73	#设置关节最大速度
	74	joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0)
	75	ret = robot.set_joint_maxvelc(joint_velc)
	76	print("robot set_joint_maxvelc ret is {0}".format(ret))
	77	#获取关节最大速度
	78	print("joint maxvelc:{0}".format(robot.get_joint_maxvelc()))
输出	241	robot set_joint_maxvelc ret is 0
	242	joint maxvelc:[1.0, 1.0, 1.0, 1.0, 1.0, 1.0]

20. set_end_max_line_acc(end_maxacc)

set_end_max_line_acc(end_maxacc)		
说明	设置直线运动最大加速度，单位 m/s ²	
示例	80	#设置直线运动最大加速度
	81	end_maxacc = 0.5
	82	ret = robot.set_end_max_line_acc(end_maxacc)
	83	print("robot set_end_max_line_acc ret is {0}".format(ret))
输出	283	robot set_end_max_line_velc ret is 0

21. get_end_max_line_acc()

get_end_max_line_acc()		
说明	获取直线运动最大加速度，单位 m/s ²	
示例	80	#设置直线运动最大加速度
	81	end_maxacc = 0.5
	82	ret = robot.set_end_max_line_acc(end_maxacc)
	83	print("robot set_end_max_line_acc ret is {0}".format(ret))
	84	#获取关节最大速度
	85	print("end_line_maxacc:{0}".format(robot.get_end_max_line_acc()))
输出	281	robot set_end_max_line_acc ret is 0
	282	end_line_maxacc:0.5

22. set_end_max_line_velc(end_maxvelc)

set_end_max_line_velc(end_maxvelc=0.1)		
说明	设置直线运动最大速度，单位：m/s	

示例	87	#设置直线运动最大速度
	88	end_maxvelc = 0.2
	89	ret = robot.set_end_max_line_velc(end_maxvelc)
	90	print("robot set_end_max_line_velc ret is {}".format(ret))
	91	#获取关节最大速度
	92	print("end_line_maxvelc:{}".format(robot.get_end_max_line_velc()))
输出	283	robot set_end_max_line_velc ret is 0

23. get_end_max_line_velc()

get_end_max_line_velc()		
说明	获取直线运动最大速度，单位 m/s	
示例	87	#设置直线运动最大速度
	88	end_maxvelc = 0.2
	89	ret = robot.set_end_max_line_velc(end_maxvelc)
	90	print("robot set_end_max_line_velc ret is {}".format(ret))
	91	#获取关节最大速度
	92	print("end_line_maxvelc:{}".format(robot.get_end_max_line_velc()))
输出	283	robot set_end_max_line_velc ret is 0
	284	end_line_maxvelc:0.2

24. set_end_max_angle_acc(end_maxacc)

set_end_max_angle_acc(end_maxacc=0.1)		
说明	设置旋转运动最大角加速度，单位 rad/s ²	
示例	94	#设置旋转运动最大角加速度
	95	end_angle_maxacc = 0.5
	96	ret = robot.set_end_max_angle_acc(end_angle_maxacc)
	97	print("robot set_end_max_angle_acc ret is {}".format(ret))
	97	print("robot set_end_max_angle_acc ret is {}".format(ret))
输出	331	robot set_end_max_angle_acc ret is 0

25. get_end_max_angle_acc()

get_end_max_angle_acc()		
说明	获取旋转运动最大角加速度，单位 rad/s ²	
示例	94	#设置旋转运动最大角加速度
	95	end_angle_maxacc = 0.5
	96	ret = robot.set_end_max_angle_acc(end_angle_maxacc)
	97	print("robot set_end_max_angle_acc ret is {}".format(ret))
	98	#获取旋转运动最大角加速度
	99	print("end_angle_maxacc:{}".format(robot.get_end_max_angle_acc()))
输出	331	robot set_end_max_angle_acc ret is 0
	332	end_angle_maxacc:0.5

26. set_end_max_angle_velc(end_maxvelc)


set_end_max_angle_velc(end_maxvelc=0.1)		
说明	设置旋转运动最大角速度，单位 rad/s	
示例	101	#设置旋转运动最大角速度
	102	end_angle_maxvelc = 0.1
	103	ret = robot.set_end_max_angle_velc(end_angle_maxvelc)
	104	print("robot set_end_max_angle_velc ret is {}".format(ret))

输出	333 robot set_end_max_angle_velc ret is 0
----	---

27. get_end_max_angle_velc()


get_end_max_angle_velc()	
说明	获取旋转运动最大角速度，单位 rad/s
示例	<pre> 101 #设置旋转运动最大角速度 102 end_angle_maxvelc = 0.1 103 ret = robot.set_end_max_angle_velc(end_angle_maxvelc) 104 print("robot set_end_max_angle_velc ret is {}".format(ret)) 105 #获取旋转运动最大角速度 106 print("end_angle_maxvelc:{}".format(robot.get_end_max_angle_velc())) </pre>
输出	<pre> 333 robot set_end_max_angle_velc ret is 0 334 end_angle_maxvelc:0.1 </pre>

28. move_to_target_in_cartesian(pos,rpy_xyz)

move_to_target_in_cartesian(pos, rpy_xyz)	
说明	<p>给出笛卡尔坐标值和欧拉角，机械臂轴动到目标位置和姿态</p> <p>pos:位置坐标 (x, y, z)，单位(m)</p> <p>rpy: 欧拉角 (rx, ry, rz),单位 (度)</p> <p>成功返回 0</p>
示例	<pre> 108 #运动至目标位置 109 pos = (-0.4,0,0) 110 rpy = (180,0,0) 111 ret = robot.move_to_target_in_cartesian(pos,rpy) 112 print("move_to_target_in_cartesian ret is:{}".format(ret)) </pre>
输出	<pre> 430 move_to_target_in_cartesian ret is:0 </pre>  <p>The screenshot displays a robotic control interface. On the left, a 3D coordinate system shows the robot's position. In the center, a table lists the current position (X: -0.400000, Y: 0.000000, Z: 0.000000) and attitude (RX: 180.000000, RY: 0.000000, RZ: 0.000000) in degrees. Below this, a '目标' (Target) dropdown menu is set to 'flange_center'. To the right, a '姿态控制' (Attitude Control) section features a diagram with arrows indicating movement directions (X+, X-, Y+, Y-, Z+, Z-). Further right, a '关节控制' (Joint Control) section lists joints 1 through 6, with 'Base' selected as the reference frame.</p>

29. move_joint(joint_radian)

move_joint(joint_radian=(0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000))	
说明	<p>轴动运动至目标位置</p> <p>传入参数为 6 个关节角，单位 rad</p> <p>成功返回 0</p>

示例	<pre> 80 #关节运动 81 joint = (0.698132,0.698132,0.698132,0.698132,0.698132,0.698132) 82 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 83 joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0) 84 robot.init_profile() 85 robot.set_joint_maxacc(joint_acc) 86 robot.set_joint_maxvelc(joint_velc) 87 ret = robot.move_joint(joint) 88 print("robot move_joint ret is {}".format(ret)) </pre>														
输出	<pre> 64 robot move_joint ret is 0 </pre>  <p>关节控制 单位(deg)</p> <table border="1"> <thead> <tr> <th>关节</th> <th>值</th> </tr> </thead> <tbody> <tr> <td>关节1</td> <td>40.000016</td> </tr> <tr> <td>关节2</td> <td>40.000016</td> </tr> <tr> <td>关节3</td> <td>40.000016</td> </tr> <tr> <td>关节4</td> <td>40.000016</td> </tr> <tr> <td>关节5</td> <td>40.000016</td> </tr> <tr> <td>关节6</td> <td>40.000016</td> </tr> </tbody> </table>	关节	值	关节1	40.000016	关节2	40.000016	关节3	40.000016	关节4	40.000016	关节5	40.000016	关节6	40.000016
关节	值														
关节1	40.000016														
关节2	40.000016														
关节3	40.000016														
关节4	40.000016														
关节5	40.000016														
关节6	40.000016														

30. move_line(joint_radian)

<pre> move_line(joint_radian=(0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000)) </pre>	
说明	<p>直线运动至目标位置</p> <p>传入参数为 6 个关节角，单位 rad</p> <p>成功返回 0</p>
示例	<pre> 117 #关节运动 118 joint = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 119 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 120 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 121 joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0) 122 robot.init_profile() 123 robot.set_joint_maxacc(joint_acc) 124 robot.set_joint_maxvelc(joint_velc) 125 ret = robot.move_joint(joint) 126 print("robot move_joint ret is {}".format(ret)) 127 128 #直线运动 129 joint = (math.radians(-17.682977),math.radians(-2.348704),math.radians(-130.588408), 130 math.radians(-38.239699),math.radians(-90),math.radians(-107.682980)) 131 end_maxacc = 0.5 132 end_maxvelc = 0.2 133 robot.init_profile() 134 robot.set_end_max_line_acc(end_maxacc) 135 robot.set_end_max_line_velc(end_maxvelc) 136 ret = robot.move_line(joint) 137 print("robot move_line ret is {}".format(ret)) </pre>
输出	<pre> 144 robot move_joint ret is 0 145 robot move_line ret is 0 </pre>

关节控制 单位(deg)	
关节1	-17.682977
关节2	-2.348704
关节3	-130.588408
关节4	-38.239705
关节5	-90.000003
关节6	-107.682980

31. move_rotate(user_coord,rotate_axis,rotate_angle)

move_rotate(user_coord, rotate_axis, rotate_angle)	
说明	<p>保持当前位置变换姿态做旋转运动</p> <pre> user_coord = {'coord_type': 2, 'calibrate_method': 0, 'calibrate_points': {"point1": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0), "point2": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0), "point3": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0)}, 'tool_desc': {"pos": (0.0, 0.0, 0.0), "ori": (1.0, 0.0, 0.0, 0.0)} } </pre> <p>rotate_axis:转轴(x,y,z) 例如: (1,0,0)表示沿 X 轴转动 rotate_angle:旋转角度 单位 (rad)</p>
示例 1	<p>1、绕着 Base 坐标系的 Z 轴旋转 20 度</p> <pre> 140 #旋转运动:绕着Base坐标系Z轴旋转20度 141 #base坐标系 142 userCoord = {'coord_type':robotcontrol.RobotCoordType.Robot_Base_Coordinate, 143 'calibrate_method':0, 144 'calibrate_points': 145 {"point1":(0,0,0,0,0,0), 146 "point2":(0,0,0,0,0,0), 147 "point3":(0,0,0,0,0,0), 148 }, 149 'tool_desc': 150 {"pos": (0.0, 0.0, 0.0), 151 "ori": (1.0, 0.0, 0.0, 0.0) 152 } 153 } 154 rotateAxis = (0,0,1) 155 rotateAngle = math.radians(20) 156 ret = robot.move_rotate(userCoord,rotateAxis,rotateAngle) 157 print("move_rotate ret is {}".format(ret)) </pre>
输出 1	298 move_rotate ret is 0
示例 2	2、绕着 user 坐标系 Y 轴转 30 度

	<pre> 171 #旋转运动:绕着User坐标系Y轴旋转30度 172 #user坐标系 173 userCoord = {'coord_type':robotcontrol.RobotCoordType.Robot_World_Coordinate, 174 'calibrate_method':robotcontrol.RobotCoordCalMethod.CoordCalMethod_xOxy, 175 'calibrate_points': 176 {'point1':(math.radians(60.443151),math.radians(42.275463),math.radians(-97.679737), 177 math.radians(-49.990510),math.radians(-90.007372),math.radians(62.567046)), 178 "point2":(math.radians(83.411541),math.radians(39.625360),math.radians(-103.796807), 179 math.radians(-53.491856),math.radians(-90.021641),math.radians(85.530279)), 180 "point3":(math.radians(81.206455),math.radians(28.381980),math.radians(-129.233955), 181 math.radians(-67.700289),math.radians(-90.019516),math.radians(83.325883)), 182 }, 183 'tool_desc': 184 {'pos': (0.0, 0.0, 0.0), 185 "ori": (1.0, 0.0, 0.0, 0.0) 186 } 187 } 188 rotateAxis = (0,1,0) 189 rotateAngle = math.radians(30) 190 ret = robot.move_rotate(userCoord,rotateAxis,rotateAngle) 191 print("move_rotate ret is {}".format(ret)) </pre>
输出 2	<pre> 323 move_rotate ret is 0 </pre>

32. clear_offline_track()

clear_offline_track()	
说明	不常用
示例	
输出	

33. append_offline_track_waypoint(waypoints)

append_offline_track_waypoint(waypoints)	
说明	不常用
示例	
输出	

34. append_offline_track_file(track_file)

append_offline_track_file(track_file)	
说明	不常用
示例	
输出	

35. startup_offline_track()

startup_offline_track()	
说明	不常用
示例	
输出	

36. stop_offline_track()

stop_offline_track()	
-----------------------------	--

说明	不常用
示例	
输出	

37. enter_tcp2canbus_mode()

enter_tcp2canbus_mode()	
说明	通知服务器进入 TCP2CANBUS 透传模式
示例	<pre> 193 #进入TCP to CANBUS透传模式 194 ret = robot.enter_tcp2canbus_mode() 195 print("enter_tcp2canbus_mode ret is {}".format(ret)) </pre>
输出	<pre> 377 enter_tcp2canbus_mode ret is 0 </pre>

38. leave_tcp2canbus_mode()

leave_tcp2canbus_mode()	
说明	通知服务器退出 TCP2CANBUS 透传模式
示例	<pre> 193 #进入TCP to CANBUS透传模式 194 ret = robot.enter_tcp2canbus_mode() 195 print("enter_tcp2canbus_mode ret is {}".format(ret)) 196 197 #退出TCP to CANBUS透传模式 198 ret = robot.leave_tcp2canbus_mode() 199 print("leave_tcp2canbus_mode ret is {}".format(ret)) </pre>
输出	<pre> 377 enter_tcp2canbus_mode ret is 0 378 leave_tcp2canbus_mode ret is 0 </pre>

39. set_waypoint_to_canbus(joint_radian)

set_waypoint_to_canbus(joint_radian=(0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000))	
说明	透传运动路点到 CANBUS 路点直接下发到 can 总线至关节驱动器，用于 ROS 开发或者机器人运动算法开发等
示例	仅供参考

	<pre> 193 #关节运动 194 joint = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 195 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 196 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 197 joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0) 198 robot.init_profile() 199 robot.set_joint_maxacc(joint_acc) 200 robot.set_joint_maxvelc(joint_velc) 201 ret = robot.move_joint(joint) 202 print("robot move_joint ret is {}".format(ret)) 203 #进入TCP to CANBUS透传模式 204 ret = robot.enter_tcp2canbus_mode() 205 print("enter_tcp2canbus_mode ret is {}".format(ret)) 206 207 i = 0 208 while(i<2000): 209 #透传运动路点到CANBUS 210 joint = (-0.308626,0.484942,-2.419299,-1.333444,-1.570796,-1.879423+i*0.000034907) 211 robot.set_waypoint_to_canbus(joint) 212 i = i + 1 213 time.sleep(0.005) 214 215 #退出TCP to CANBUS透传模式 216 ret = robot.leave_tcp2canbus_mode() 217 print("leave_tcp2canbus_mode ret is {}".format(ret)) </pre>
输出	<div> <div>关节控制 单位(deg)</div> <div> <div>关节1</div> <div>-</div> <div>-17.682967</div> </div> <div> <div>关节2</div> <div>-</div> <div>27.785129</div> </div> <div> <div>关节3</div> <div>-</div> <div>-138.615616</div> </div> <div> <div>关节4</div> <div>-</div> <div>-76.400713</div> </div> <div> <div>关节5</div> <div>-</div> <div>-89.999982</div> </div> <div> <div>关节6</div> <div>-</div> <div>-103.744959</div> </div> </div>

40. remove_all_waypoint()

remove_all_waypoint()		
说明	清除所有已经设置的全局路点 用于新的轨迹运动之前，清除之前的全局路点	
示例	<pre> 223 #清除所有已经设置的全局路点 224 ret = robot.remove_all_waypoint() 225 print("remove_all_waypoint ret is {}".format(ret)) </pre>	
输出	9082	remove_all_waypoint ret is 0

41. add_waypoint(joint_radian)

add_waypoint(joint_radian=(0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000))		
说明	添加全局路点用于轨迹运动 单位 rad	
示例	<pre> 245 #添加3个全局路点 246 wp1 = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 247 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 248 wp2 = (math.radians(-17.682979),math.radians(-0.337404),math.radians(-131.976125), 249 math.radians(-41.638725),math.radians(-90),math.radians(-107.682980)) 250 wp3 = (math.radians(-34.376667),math.radians(2.484861),math.radians(-129.077134), 251 math.radians(-41.562015),math.radians(-90),math.radians(-124.376707)) 252 robot.add_waypoint(wp1) 253 robot.add_waypoint(wp2) 254 robot.add_waypoint(wp3) </pre>	
输出		

42. set_blend_radius(blend_radius)效果不明显

set_blend_radius(blend_radius=0.01)		
说明	设置轨迹运动中 moveP 的交融半径 单位 m, 范围[0.01,0.05]	
示例	244	#设置交融半径, 0.01
	245	ret = robot.set_blend_radius(0.01)
	246	print("set_blend_radius ret is {}".format(ret))
输出	455	set_blend_radius ret is 0

43. set_circular_loop_times(circular_count)

set_circular_loop_times(circular_count=1)		
说明	设置圆运动圈数 当 circular_count 大于 0 时, 机械臂进行圆运动 circular_count 次 当 circular_count 等于 0 时, 机械臂进行圆弧轨迹运动	
示例	256	#设置运动圈数, 圈数等于2
	257	robot.set_circular_loop_times(2)
输出		

44. set_user_coord(user_coord)用途

set_user_coord(user_coord)		
说明	设置用户坐标系	
示例		
输出		

45. set_base_coord()用途


set_base_coord()		
说明	设置 Base 坐标系	
示例		
输出		

46. check_user_coord(user_coord)无效

check_user_coord(user_coord)		
说明	检查用户坐标系参数设置是否合理 user_coord = {'coord_type': 2, 'calibrate_method': 0, 'calibrate_points': {"point1": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0), "point2": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0), "point3": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0)},	

	<pre> 'tool_desc': {"pos": (0.0, 0.0, 0.0), "ori": (1.0, 0.0, 0.0, 0.0)} } </pre>
示例	<pre> 272 #user坐标系 273 userCoord = {'coord_type':robotcontrol.RobotCoordType.Robot_World_Coordinate, 274 'calibrate_method':robotcontrol.RobotCoordCalMethod.CoordCalMethod_xOxy, 275 'calibrate_points': 276 {"point1":(math.radians(60.443151),math.radians(42.275463),math.radians(-97.679737), 277 math.radians(-49.990510),math.radians(-90.007372),math.radians(62.567046)), 278 "point2":(math.radians(83.411541),math.radians(39.625360),math.radians(-103.796807), 279 math.radians(-53.491856),math.radians(-90.021641),math.radians(85.530279)), 280 "point3":(math.radians(81.206455),math.radians(28.381980),math.radians(-129.233955), 281 math.radians(-67.700289),math.radians(-90.019516),math.radians(83.325883)), 282 }, 283 'tool_desc': 284 {"pos": (0.0, 0.0, 0.0), 285 "ori": (1.0, 0.0, 0.0, 0.0)} 286 } 287 288 #检查用户坐标系是否合理 289 ret = robot.check_user_coord(userCoord) 290 print("check_user_coord ret is {}".format(ret)) </pre>
输出	<pre> 571 check_user_coord ret is 0 </pre>

47. set_relative_offset_on_base(relative_pos, relative_ori)

set_relative_offset_on_base(relative_pos, relative_ori)	
说明	<p>设置基于基座坐标系运动偏移量</p> <p>relative_pos=(x, y, z) 相对位移, 单位(m)</p> <p>relative_ori=(w,x,y,z) 目标姿态</p>
示例	<pre> 292 ## #关节运动 293 joint = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 294 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 295 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 296 joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0) 297 robot.init_profile() 298 robot.set_joint_maxacc(joint_acc) 299 robot.set_joint_maxvelc(joint_velc) 300 ret = robot.move_joint(joint) 301 print("robot move_joint ret is {}".format(ret)) 302 #基座坐标系下相对偏移 303 pos_off = (0,0,0.1) 304 rpy_off = (0,0,0) 305 ori_off = robot.rpy_to_quaternion(rpy_off) 306 ret = robot.set_relative_offset_on_base(pos_off,ori_off) 307 print("set_relative_offset_on_base ret is {}".format(ret)) 308 ret = robot.move_joint(joint) 309 print("robot move_joint ret is {}".format(ret)) </pre>
输出	<pre> 72 robot move_joint ret is 0 73 set_relative_offset_on_base ret is 0 74 robot move_joint ret is 0 </pre> <div>  </div>

48. set_relative_offset_on_user(relative_pos, relative_ori, user_coord)

set_relative_offset_on_user()		
说明	<p>设置基于用户标系运动偏移量</p> <p>relative_pos=(x, y, z) 相对位移, 单位(m)</p> <p>relative_ori=(w,x,y,z) 目标姿态</p> <p>user_coord:用户坐标系</p> <pre>user_coord = {'coord_type': 2, 'calibrate_method': 0, 'calibrate_points': {"point1": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0), "point2": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0), "point3": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0)}, 'tool_desc': {"pos": (0.0, 0.0, 0.0), "ori": (1.0, 0.0, 0.0, 0.0)}}</pre>	
示例	<p>User 坐标系下沿 Z 轴平移 0.1m</p> <pre>1 312 #关节运动 313 joint = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 314 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 315 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 316 joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0) 317 robot.init_profile() 318 robot.set_joint_maxacc(joint_acc) 319 robot.set_joint_maxvelc(joint_velc) 320 ret = robot.move_joint(joint) 321 print("robot move_joint ret is {}".format(ret)) 322 323 ## 324 userCoord = {'coord_type':robotcontrol.RobotCoordType.Robot_World_Coordinate, 325 'calibrate_method':robotcontrol.RobotCoordCalMethod.CoordCalMethod_xOxy, 326 'calibrate_points': 327 {"point1":(math.radians(60.443151),math.radians(42.275463),math.radians(-97.679737), 328 math.radians(-49.990510),math.radians(-90.007372),math.radians(62.567046)), 329 "point2":(math.radians(83.411541),math.radians(39.625360),math.radians(-103.796807), 330 math.radians(-53.491856),math.radians(-90.021641),math.radians(85.530279)), 331 "point3":(math.radians(81.206455),math.radians(28.381980),math.radians(-129.233955), 332 math.radians(-67.700289),math.radians(-90.019516),math.radians(83.325883)), 333 }, 334 'tool_desc': 335 {"pos": (0.0, 0.0, 0.0), 336 "ori": (1.0, 0.0, 0.0, 0.0) 337 } 338 } 339 340 #基座标系下相对偏移 341 pos_off = (0,0.1,0) 342 rpy_off = (0,0,0) 343 ori_off = robot.rpy_to_quaternion(rpy_off) 344 ret = robot.set_relative_offset_on_user(pos_off,ori_off,userCoord) 345 print("set_relative_offset_on_user ret is {}".format(ret)) 346 ret = robot.move_joint(joint) 347 print("robot move_joint ret is {}".format(ret))</pre>	
输出	197	robot move_joint ret is 0
1	198	set_relative_offset_on_user ret is 0
	199	robot move_joint ret is 0

沿工具坐标系 Z 轴旋转 30 度

```
350 ##          #end坐标系
351         userCoord = {'coord_type':robotcontrol.RobotCoordType.Robot_End_Coordinate,
352                       'calibrate_method':0,
353                       'calibrate_points':
354                           {"point1":(0,0,0,0,0,0),
355                             "point2":(0,0,0,0,0,0),
356                             "point3":(0,0,0,0,0,0),
357                           },
358                       'tool_desc':
359                           {"pos": (0.0, 0.0, 0.0),
360                             "ori": (1.0, 0.0, 0.0, 0.0)
361                           }
362             }
363
364         #User座标系下相对偏移
365         pos_off = (0,0,0)
366         rpy_off = (0,0,math.radians(30))
367         ori_off = robot.rpy_to_quaternion(rpy_off)
368         ret = robot.set_relative_offset_on_user(pos_off,ori_off,userCoord)
369         print("set_relative_offset_on_user ret is {}".format(ret))
370         ret = robot.move_joint(joint)
371         print("robot move_joint ret is {}".format(ret))
```

位置(m)	X: -0.400000	Y: 0.000000	Z: 0.000000
姿态(deg)	RX: 180.000000	RY: 0.000000	RZ: -29.999981
关节状态 单位(deg)			
关节1	-17.682979	关节2	27.785112
关节3	-138.615629	关节4	-76.400741
关节5	-90.000003	关节6	-77.682979

49. set_no_arrival_ahead()

set_no_arrival_ahead()	
说明	取消提前到位设置
示例	<pre> 349 wp1 = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 350 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 351 wp2 = (math.radians(-17.682979),math.radians(-0.337404),math.radians(-131.976125), 352 math.radians(-41.638725),math.radians(-90),math.radians(-107.682980)) 353 wp3 = (math.radians(-34.376667),math.radians(2.484861),math.radians(-129.077134), 354 math.radians(-41.562015),math.radians(-90),math.radians(-124.376707)) 355 #关节运动到wp1 356 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 357 joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0) 358 robot.init_profile() 359 robot.set_joint_maxacc(joint_acc) 360 robot.set_joint_maxvelc(joint_velc) 361 robot.set_end_max_line_acc(0.5) 362 robot.set_end_max_line_velc(0.5) 363 ret = robot.move_joint(wp1) 364 print("robot move_joint ret is {}".format(ret)) 365 366 #无提前到位 367 robot.set_no_arrival_ahead() 368 ret = robot.move_joint(wp2) 369 print("robot move_joint ret is {}".format(ret)) 370 ret = robot.move_joint(wp3) 371 print("robot move_joint ret is {}".format(ret)) </pre>

输出	机器人会在 wp2 点停顿
----	---------------

50. set_arrival_ahead_distance(distance)

set_arrival_ahead_distance(distance=0.0)	
说明	<p>设置距离模式下的提前到位距离</p> <p>distance 提前到位距离 单位（米），只能用于 move_joint()</p>
示例	<pre> 349 wp1 = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 350 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 351 wp2 = (math.radians(-17.682979),math.radians(-0.337404),math.radians(-131.976125), 352 math.radians(-41.638725),math.radians(-90),math.radians(-107.682980)) 353 wp3 = (math.radians(-34.376667),math.radians(2.484861),math.radians(-129.077134), 354 math.radians(-41.562015),math.radians(-90),math.radians(-124.376707)) 355 356 #关节运动到wp1 357 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 358 joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0) 359 robot.init_profile() 360 robot.set_joint_maxacc(joint_acc) 361 robot.set_joint_maxvelc(joint_velc) 362 robot.set_end_max_line_acc(0.5) 363 robot.set_end_max_line_velc(0.5) 364 ret = robot.move_joint(wp1) 365 print("robot move_joint ret is {}".format(ret)) 366 367 #提前到位距离模式 368 robot.set_arrival_ahead_distance(0.05) 369 ret = robot.move_joint(wp2) 370 print("robot move_joint ret is {}".format(ret)) 371 ret = robot.move_joint(wp3) 372 print("robot move_joint ret is {}".format(ret)) 373 robot.set_no_arrival_ahead() 374 ret = robot.move_joint(wp1) 375 print("robot move_joint ret is {}".format(ret)) </pre>
输出	机械臂在 wp2 和 wp3 不停顿，起到平滑过渡的效果

51. set_arrival_ahead_time(sec)

set_arrival_ahead_time(sec=0.0)	
说明	<p>设置时间模式下的提前到位时间</p> <p>sec 提前到位时间 单位（秒），只适用于 move_joint()</p>
示例	<pre> 349 wp1 = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 350 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 351 wp2 = (math.radians(-17.682979),math.radians(-0.337404),math.radians(-131.976125), 352 math.radians(-41.638725),math.radians(-90),math.radians(-107.682980)) 353 wp3 = (math.radians(-34.376667),math.radians(2.484861),math.radians(-129.077134), 354 math.radians(-41.562015),math.radians(-90),math.radians(-124.376707)) 355 356 #关节运动到wp1 357 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 358 joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0) 359 robot.init_profile() 360 robot.set_joint_maxacc(joint_acc) 361 robot.set_joint_maxvelc(joint_velc) 362 robot.set_end_max_line_acc(0.5) 363 robot.set_end_max_line_velc(0.5) 364 ret = robot.move_joint(wp1) 365 print("robot move_joint ret is {}".format(ret)) 366 367 #提前到位时间模式 368 robot.set_arrival_ahead_time(0.5) 369 ret = robot.move_joint(wp2) 370 print("robot move_joint ret is {}".format(ret)) 371 ret = robot.move_joint(wp3) 372 print("robot move_joint ret is {}".format(ret)) 373 robot.set_no_arrival_ahead() 374 ret = robot.move_joint(wp1) 375 print("robot move_joint ret is {}".format(ret)) </pre>
输出	机械臂在 wp2 和 wp3 不停顿，起到平滑过渡效果

52. set_arrival_ahead_blend(distance)

set_arrival_ahead_blend(distance=0.0)	
说明	设置距离模式下交融半径距离 blend 交融半径 单位（米），适用于任何类型的运动
示例	<pre>349 wp1 = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 350 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 351 wp2 = (math.radians(-17.682979),math.radians(-0.337404),math.radians(-131.976125), 352 math.radians(-41.638725),math.radians(-90),math.radians(-107.682980)) 353 wp3 = (math.radians(-34.376667),math.radians(2.484861),math.radians(-129.077134), 354 math.radians(-41.562015),math.radians(-90),math.radians(-124.376707)) 355 #关节运动到wp1 356 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 357 joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0) 358 robot.init_profile() 359 robot.set_joint_maxacc(joint_acc) 360 robot.set_joint_maxvelc(joint_velc) 361 robot.set_end_max_line_acc(0.5) 362 robot.set_end_max_line_velc(0.5) 363 ret = robot.move_joint(wp1) 364 print("robot move_joint ret is {}".format(ret)) 365 366 #提前到位交融半径模式 367 robot.set_arrival_ahead_blend(0.05) 368 ret = robot.move_joint(wp2) 369 print("robot move_joint ret is {}".format(ret)) 370 ret = robot.move_line(wp3) 371 print("robot move_joint ret is {}".format(ret)) 372 robot.set_no_arrival_ahead() 373 ret = robot.move_joint(wp1) 374 print("robot move_joint ret is {}".format(ret))</pre>
输出	机械臂在 wp2 和 wp3 不停顿，起到平滑效果

53. move_track(track)

move_track(track)	
说明	轨迹运动，包括： 圆弧运动 RobotMoveTrackType.ARC_CIR 轨迹运动 RobotMoveTrackType.CARTESIAN_MOVEP
示例 1	圆运动

	<pre>221 #轨迹运动测试 222 223 #清除所有已经设置的全局路点 224 ret = robot.remove_all_waypoint() 225 print("remove_all_waypoint ret is {}".format(ret)) 226 227 #关节运动 228 joint = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 229 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 230 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 231 joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0) 232 robot.init_profile() 233 robot.set_joint_maxacc(joint_acc) 234 robot.set_joint_maxvelc(joint_velc) 235 ret = robot.move_joint(joint) 236 print("robot move_joint ret is {}".format(ret)) 237 238 #设置圆弧运动速度加速度 239 end_maxacc = 0.5 240 end_maxvelc = 0.2 241 robot.init_profile() 242 robot.set_end_max_line_acc(end_maxacc) 243 robot.set_end_max_line_velc(end_maxvelc) 244 245 #添加3个全局路点 246 wp1 = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 247 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 248 wp2 = (math.radians(-17.682979),math.radians(-0.337404),math.radians(-131.976125), 249 math.radians(-41.638725),math.radians(-90),math.radians(-107.682980)) 250 wp3 = (math.radians(-34.376667),math.radians(2.484861),math.radians(-129.077134), 251 math.radians(-41.562015),math.radians(-90),math.radians(-124.376707)) 252 robot.add_waypoint(wp1) 253 robot.add_waypoint(wp2) 254 robot.add_waypoint(wp3) 255 256 #设置运动圈数,圈数等于2 257 robot.set_circular_loop_times(2) 258 259 #圆弧运动 260 ret = robot.move_track(robotcontrol.RobotMoveTrackType.ARC_CIR) 261 print("move_track ret is {}".format(ret))</pre>
输出 1	9084 move_track ret is 0
示例 2	圆弧运动

	<pre>221 #轨迹运动测试 222 223 #清除所有已经设置的全局路点 224 ret = robot.remove_all_waypoint() 225 print("remove_all_waypoint ret is {}".format(ret)) 226 227 #关节运动 228 joint = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 229 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 230 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 231 joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0) 232 robot.init_profile() 233 robot.set_joint_maxacc(joint_acc) 234 robot.set_joint_maxvelc(joint_velc) 235 ret = robot.move_joint(joint) 236 print("robot move_joint ret is {}".format(ret)) 237 238 #设置圆弧运动速度加速度 239 end_maxacc = 0.5 240 end_maxvelc = 0.2 241 robot.init_profile() 242 robot.set_end_max_line_acc(end_maxacc) 243 robot.set_end_max_line_velc(end_maxvelc) 244 245 #添加3个全局路点 246 wp1 = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 247 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 248 wp2 = (math.radians(-17.682979),math.radians(-0.337404),math.radians(-131.976125), 249 math.radians(-41.638725),math.radians(-90),math.radians(-107.682980)) 250 wp3 = (math.radians(-34.376667),math.radians(2.484861),math.radians(-129.077134), 251 math.radians(-41.562015),math.radians(-90),math.radians(-124.376707)) 252 robot.add_waypoint(wp1) 253 robot.add_waypoint(wp2) 254 robot.add_waypoint(wp3) 255 256 #设置运动圈数,圈数等于0 257 robot.set_circular_loop_times(0) 258 259 #圆运动 260 ret = robot.move_track(robotcontrol.RobotMoveTrackType.ARC_CIR) 261 print("move_track ret is {}".format(ret))</pre>
输出 2	9084 move_track ret is 0
示例 3	moveP

	<pre> 221 #轨迹运动测试 222 223 #清除所有已经设置的全局路点 224 ret = robot.remove_all_waypoint() 225 print("remove_all_waypoint ret is {}".format(ret)) 226 227 #关节运动 228 joint = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 229 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 230 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 231 joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0) 232 robot.init_profile() 233 robot.set_joint_maxacc(joint_acc) 234 robot.set_joint_maxvelc(joint_velc) 235 ret = robot.move_joint(joint) 236 print("robot move_joint ret is {}".format(ret)) 237 238 #设置圆弧运动速度加速度 239 end_maxacc = 1 240 end_maxvelc = 1 241 robot.init_profile() 242 robot.set_end_max_line_acc(end_maxacc) 243 robot.set_end_max_line_velc(end_maxvelc) 244 #设置交融半径，0.04 245 robot.set_blend_radius(0.04) 246 247 #添加3个全局路点 248 wp1 = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 249 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 250 wp2 = (math.radians(-17.682979),math.radians(-0.337404),math.radians(-131.976125), 251 math.radians(-41.638725),math.radians(-90),math.radians(-107.682980)) 252 wp3 = (math.radians(-34.376667),math.radians(2.484861),math.radians(-129.077134), 253 math.radians(-41.562015),math.radians(-90),math.radians(-124.376707)) 254 robot.add_waypoint(wp1) 255 robot.add_waypoint(wp2) 256 robot.add_waypoint(wp3) 257 258 #moveP测试 259 ret = robot.move_track(robotcontrol.RobotMoveTrackType.CARTESIAN_MOVEP) 260 print("move_track ret is {}".format(ret)) </pre>
输出 3	430 move_track ret is 0


54. forward_kin(joint_radian)

forward_kin(joint_radian=(0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000))	
说明	<p>正解，通过六个关节角求出法兰坐标系在 Base 下的位置和姿态</p> <p>oint_radian: 六个关节的关节角，单位(rad)</p> <p>返回值:</p> <p>‘pos’: [x,y,z], 单位 m</p> <p>‘ori’: [w,x,y,z], 四元数</p>
示例	<pre> 408 #正解FK 409 joint = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 410 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 411 ret = robot.forward_kin(joint) 412 #打印位置XYZ 413 print("pos_x = {:.6f}".format(ret['pos'][0])) 414 print("pos_y = {:.6f}".format(ret['pos'][1])) 415 print("pos_z = {:.6f}".format(ret['pos'][2])) 416 417 rpy = robot.quaternion_to_rpy(ret['ori']) 418 #打印姿态RPY 419 print("RX = {:.6f}".format(math.degrees(rpy[0]))) 420 print("RY = {:.6f}".format(math.degrees(rpy[1]))) 421 print("RZ = {:.6f}".format(math.degrees(rpy[2]))) </pre>

输出	<pre> 1002 pos_x = -0.400000 1003 pos_y = -0.000000 1004 pos_z = -0.000000 1005 RX = 180.000000 1006 RY = -0.000007 1007 RZ = 0.000003 </pre> 
----	--

55. inverse_kin(joint_radian,pos,ori)


inverse_kin(joint_radian=(0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000), pos=(0.0, 0.0, 0.0), ori=(1.0, 0.0, 0.0, 0.0))


说明	<p>逆解，根据给定的位置和姿态，求出对应的六个关节角</p> <p>joint_radian:起始点六个关节的关节角，单位(rad)，作为参考点</p> <p>pos 位置(x, y, z)单位(m)</p> <p>ori 位姿(w, x, y, z)</p> <p>成功返回六个关节角</p>
示例	<pre> 423 #逆解IK 424 joint = (math.radians(-17.682977),math.radians(27.785112),math.radians(-138.615629), 425 math.radians(-76.400734),math.radians(-90),math.radians(-107.682980)) 426 pos_target = (0.4,0,0.1) 427 rpy_target = (math.radians(180),math.radians(0),math.radians(0)) 428 ori_target = robot.rpy_to_quaternion(rpy_target) 429 ret = robot.inverse_kin(joint,pos_target,ori_target) 430 #打印目标关节角 431 print("target joint = {0}",ret) 432 #轴动运动至目标点位 433 joint_acc = (2.0,2.0,2.0,2.0,2.0,2.0) 434 joint_velc = (1.0,1.0,1.0,1.0,1.0,1.0) 435 robot.init_profile() 436 robot.set_joint_maxacc(joint_acc) 437 robot.set_joint_maxvelc(joint_velc) 438 ret = robot.move_joint(ret['joint']) 439 print("robot move_joint ret is {0}".format(ret)) </pre>
输出	<p>运动到位后，坐标显示如下：</p> 

56. base_to_user(pos, ori, user_coord, user_tool)

base_to_user(pos, ori, user_coord, user_tool)

说明	<p>Base 坐标系坐标转 User 坐标系坐标</p> <p>pos:基座标系下的位置(x, y, z)单位(m)</p> <p>ori:基座标系下的姿态(w, x, y, z)</p> <p>user_coord:用户坐标系</p> <pre> user_coord = {'coord_type': 2, 'calibrate_method': 0, 'calibrate_points': {"point1": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0), </pre>
----	---

	<pre> "point2": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0), "point3": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0)}, 'tool_desc': { "pos": (0.0, 0.0, 0.0), "ori": (1.0, 0.0, 0.0, 0.0, 0.0)} } } user_tool 用户工具描述 user_tool={"pos": (x, y, z), "ori": (w, x, y, z)} 成功返回：切换坐标系后的坐标，详见示例 </pre>
示例 1	<p>flange_center 在 Base 下的坐标转 flange_center 在 userCoord(tray22) 下的坐标</p> <pre> 441 #flange_center在Base下的坐标转flange_center在userCoord(tray22)下的坐标 442 pos_base = (-0.4,0,0) 443 rpy_base = (math.radians(180),math.radians(0),math.radians(0)) 444 ori_base = robot.rpy_to_quaternion(rpy_base) 445 446 userCoord = {'coord_type':robotcontrol.RobotCoordType.Robot_World_Coordinate, 447 'calibrate_method':robotcontrol.RobotCoordCalMethod.CoordCalMethod_x0xy, 448 'calibrate_points': 449 { 450 "point1":(math.radians(60.443151),math.radians(42.275463),math.radians(-97.679737), 451 math.radians(-49.990510),math.radians(-90.007372),math.radians(62.567046)), 452 "point2":(math.radians(83.411541),math.radians(39.625360),math.radians(-103.796807), 453 math.radians(-53.491856),math.radians(-90.021641),math.radians(85.530279)), 454 "point3":(math.radians(81.206455),math.radians(28.381980),math.radians(-129.233955), 455 math.radians(-67.700289),math.radians(-90.019516),math.radians(83.325883)) 456 }, 457 'tool_desc': 458 { 459 "pos": (0.0, 0.0, 0.0), 460 "ori": (1.0, 0.0, 0.0, 0.0, 0.0) 461 } 462 } 463 userTool = {'pos':(0,0,0),'ori':(1,0,0,0)} 464 465 ret = robot.base_to_user(pos_base,ori_base,userCoord,userTool) 466 print("F_B to F_U ret = {}".format(ret)) 467 ## 打印user坐标系下的位置坐标 468 print("X_user = {:.6f}".format(ret['pos'][0])) 469 print("Y_user = {:.6f}".format(ret['pos'][1])) 470 print("Z_user = {:.6f}".format(ret['pos'][2])) 471 472 rpy_user = robot.quaternion_to_rpy(ret['ori']) 473 #打印user坐标系下的姿态 474 print("RX_user = {:.6f}".format(math.degrees(rpy_user[0]))) 475 print("RY_user = {:.6f}".format(math.degrees(rpy_user[1]))) 476 print("RZ_user = {:.6f}".format(math.degrees(rpy_user[2]))) 477 478 ... </pre>
输出 1	 <pre> 236 X_user = -0.196999 237 Y_user = 0.599515 238 Z_user = -0.015776 239 RX_user = -179.602783 240 RY_user = -0.376928 241 RZ_user = -0.312728 </pre>
示例 2	
输出 2	
示例 3	<p>flange_center 在 Base 下的坐标转 tcp1(0,0,0.45)在 userCoord(tray22) 下的坐标</p>


	<pre> 476 #flange_center在Base下的坐标转tcp1(0,0,0.45)在userCoord(tray22)下的坐标 477 #在Base下坐标为: (-0.4,0,0) (180,0,0) 478 pos_base = (-0.4,0,0) 479 rpy_base = (math.radians(180),math.radians(0),math.radians(0)) 480 ori_base = robot.rpy_to_quaternion(rpy_base) 481 482 userCoord = {'coord_type': robotcontrol.RobotCoordType.Robot_World_Coordinate, 483 'calibrate_method': robotcontrol.RobotCoordCalMethod.CoordCalMethod_xOxy, 484 'calibrate_points': 485 {"point1": (math.radians(60.443151),math.radians(42.275463),math.radians(-97.679737), 486 math.radians(-49.990510),math.radians(-90.007372),math.radians(62.567046)), 487 "point2": (math.radians(83.411541),math.radians(39.625360),math.radians(-103.796807), 488 math.radians(-53.491856),math.radians(-90.021641),math.radians(85.530279)), 489 "point3": (math.radians(81.206455),math.radians(28.381980),math.radians(-129.233955), 490 math.radians(-67.700289),math.radians(-90.019516),math.radians(83.325883)) 491 }, 492 'tool_desc': 493 {"pos": (0.0, 0.0, 0.0), 494 "ori": (1.0, 0.0, 0.0, 0.0)} 495 } 496 497 userTool = {'pos': (0,0,0.45), 'ori': (1,0,0,0)} 498 499 ret = robot.base_to_user(pos_base,ori_base,userCoord,userTool) 500 ## print("F_B to F_U ret = {}".format(ret)) 501 #打印user坐标系下的位置坐标 502 print("X_user = {:.6f}".format(ret['pos'][0])) 503 print("Y_user = {:.6f}".format(ret['pos'][1])) 504 print("Z_user = {:.6f}".format(ret['pos'][2])) 505 506 rpy_user = robot.quaternion_to_rpy(ret['ori']) 507 #打印user坐标系下的姿态 508 print("RX_user = {:.6f}".format(math.degrees(rpy_user[0]))) 509 print("RY_user = {:.6f}".format(math.degrees(rpy_user[1]))) 510 print("RZ_user = {:.6f}".format(math.degrees(rpy_user[2]))) 511 ... </pre>
输出 3	<div> <pre> 371 X_user = -0.194022 372 Y_user = 0.602618 373 Z_user = -0.465755 374 RX_user = -179.602783 375 RY_user = -0.376928 376 RZ_user = -0.312728 </pre>  </div>

57. user_to_base(pos, ori, user_coord, user_tool)无效

user_to_base(pos, ori, user_coord, user_tool)	
说明	<p>User 坐标系坐标转 Base 坐标系坐标</p> <p>pos:user 坐标系下的位置(x, y, z)单位(m)</p> <p>ori:Base 坐标系下的姿态(w, x, y, z)</p> <p>user_coord:用户坐标系</p> <pre> user_coord = {'coord_type': 2, 'calibrate_method': 0, 'calibrate_points': {"point1": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0), "point2": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0), "point3": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0)}, 'tool_desc': {"pos": (0.0, 0.0, 0.0), "ori": (1.0, 0.0, 0.0, 0.0)} } </pre> <p>user_tool 用户工具描述</p> <pre> user_tool={"pos": (x, y, z), "ori": (w, x, y, z)} </pre> <p>成功返回: 切换坐标系后的坐标, 详见示例</p>

示例	flange_center 在 UserCoord(tray22)下的坐标转 flange_center 在 Base 下的坐标。
输出	

58. base_to_base_additional_tool(flange_pos, flange_ori, user_tool)

base_to_base_additional_tool(flange_pos, flange_ori, user_tool)	
说明	<p>pos:基于基座标系的法兰盘中心位置信息(x, y, z)单位(m)</p> <p>ori:基于基座标系的姿态信息(w, x, y, z)</p> <p>user_tool 用户工具描述</p> <p>user_tool={"pos": (x, y, z), "ori": (w, x, y, z)}</p> <p>成功返回: 基于基座标系的工具末端位置位置和姿态信息{"pos": (x, y, z), "ori": (w, x, y, z)}</p>
示例	<p>工具参数为(0.45,0,0) (1,0,0,0)</p> <pre> 621 #base to base additional tool 622 #flange_center在Base下的坐标为(-0.4,0,0) (-180,0,0) 623 flange_pos = (-0.4,0,0) 624 flange_rpy = (math.radians(180),math.radians(0),math.radians(0)) 625 #工具参数为(0,0,0.45) (1,0,0,0) 626 flange_ori = robot.rpy_to_quaternion(flange_rpy) 627 userTool = {'pos':(0,0,0.45),'ori':(1,0,0,0)} 628 629 ret = robot.base_to_base_additional_tool(flange_pos,flange_ori,userTool) 630 #打印位置XYZ 631 print("base_additional_tool X = {:.6f}".format(ret['pos'][0])) 632 print("base_additional_tool Y = {:.6f}".format(ret['pos'][1])) 633 print("base_additional_tool Z = {:.6f}".format(ret['pos'][2])) 634 635 additional_tool_rpy = robot.quaternion_to_rpy(ret['ori']) 636 #打印姿态RPY 637 print("base_additional_tool RX = {:.6f}".format(math.degrees(additional_tool_rpy[0]))) 638 print("base_additional_tool RY = {:.6f}".format(math.degrees(additional_tool_rpy[1]))) 639 print("base_additional_tool RZ = {:.6f}".format(math.degrees(additional_tool_rpy[2]))) </pre>
输出	<pre> 396 base_additional_tool X = -0.400000 397 base_additional_tool Y = 0.000000 398 base_additional_tool Z = -0.450000 399 base_additional_tool RX = -180.000000 400 base_additional_tool RY = -0.000000 401 base_additional_tool RZ = 0.000000 </pre> 

59. rpy_to_quaternion(rpy)

rpy_to_quaternion(rpy)	
说明	<p>欧拉角转四元数</p> <p>rpy:欧拉角(rx, ry, rz), 单位(rad)</p> <p>成功返回: 四元数结果</p>

示例	<pre> 642 #欧拉角转四元数 643 #欧拉角为(0,0,0) 644 rpy = (math.radians(0),math.radians(0),math.radians(0)) 645 ori = robot.rpy_to_quaternion(rpy) 646 print("ori.w = {:.6f}".format(ori[0])) 647 print("ori.x = {:.6f}".format(ori[1])) 648 print("ori.y = {:.6f}".format(ori[2])) 649 print("ori.z = {:.6f}".format(ori[3])) </pre>
输出	<pre> 60 ori.w = 1.000000 61 ori.x = 0.000000 62 ori.y = 0.000000 63 ori.z = 0.000000 </pre>

60. quaternion_to_rpy(ori)

quaternion_to_rpy(ori)	
说明	<p>四元数转欧拉角</p> <p>四元数(w, x, y, z)</p> <p>成功返回：欧拉角结果</p>
示例	<pre> 642 #欧拉角转四元数 643 #欧拉角为(0,0,0) 644 rpy = (math.radians(0),math.radians(0),math.radians(0)) 645 ori = robot.rpy_to_quaternion(rpy) 646 print("ori.w = {:.6f}".format(ori[0])) 647 print("ori.x = {:.6f}".format(ori[1])) 648 print("ori.y = {:.6f}".format(ori[2])) 649 print("ori.z = {:.6f}".format(ori[3])) 650 651 #四元数转欧拉角 652 rpy_1 = robot.quaternion_to_rpy(ori) 653 print("RX = {:.6f}".format(rpy_1[0])) 654 print("RY = {:.6f}".format(rpy_1[1])) 655 print("RZ = {:.6f}".format(rpy_1[2])) </pre>
输出	<pre> 97 ori.w = 1.000000 98 ori.x = 0.000000 99 ori.y = 0.000000 100 ori.z = 0.000000 101 RX = 0.000000 102 RY = -0.000000 103 RZ = 0.000000 </pre>

61. set_tool_end_param(tool_end_param)用途

set_tool_end_param(tool_end_param)	
说明	<p>设置末端工具参数</p> <p>末端工具参数： tool_end_param={"pos": (x, y, z), "ori": (w, x, y, z)}</p>

示例	<pre> 653 #设置末端工具参数 654 tool_end_param = {"pos":(0,0.2,0.1),"ori":(1,0,0,0)} 655 ret = robot.set_tool_end_param(tool_end_param) 656 print("set_tool_end_param ret is {}".format(ret)) </pre>
输出	116 set_tool_end_param ret is 0

62. set_none_tool_dynamics_param()

set_none_tool_dynamics_param()	
说明	设置无工具的动力学参数
示例	<pre> 661 #设置无工具的动力学参数 662 ret = robot.set_none_tool_dynamics_param() 663 print("set_none_tool_dynamics_param ret is {}".format(ret)) </pre>
输出	145 set_none_tool_dynamics_param ret is 0

63. set_tool_dynamics_param(tool_dynamics)

set_tool_dynamics_param(tool_dynamics)	
说明	<p>设置工具的动力学参数</p> <p>tool_dynamics:运动学参数</p> <p>tool_dynamics = 位置, 单位(m) : {"position": (0.0, 0.0, 0.0), 负载, 单位(kg): "payload": 1.0, 惯量: "inertia": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0)}</p>
示例	<pre> 665 #设置工具的动力学参数 666 tool_set_dyn = {"position":(0,0,0.05), 667 "payload":1.0, 668 "inertia":(0,0,0,0,0,0)} 669 ret = robot.set_tool_dynamics_param(tool_set_dyn) 670 print("set_tool_dynamics_param ret is {}".format(ret)) </pre>
输出	174 set_tool_dynamics_param ret is 0

64. get_tool_dynamics_param()单位

get_tool_dynamics_param()	
说明	<p>获取末端工具动力学参数</p> <p>需要连接真实机械臂, 数据才正确</p> <p>成功返回: 运动学参数</p> <p>tool_dynamics = 位置, 单位(mm) : {"position": (0.0, 0.0, 0.0), 负载, 单位(kg): "payload": 1.0, 惯量: "inertia": (0.0, 0.0, 0.0, 0.0, 0.0, 0.0)}</p>

示例	<pre> 665 #设置工具的动力学参数 666 tool_set_dyn = {"position":(0,0,0.05), 667 "payload":1.0, 668 "inertia":(0,0,0,0,0,0)} 669 ret = robot.set_tool_dynamics_param(tool_set_dyn) 670 print("set_tool_dynamics_param ret is {}".format(ret)) 671 672 #获取工具动力学参数 673 tool_dyn = robot.get_tool_dynamics_param() 674 print(tool_dyn) </pre>
输出	<pre> 174 set_tool_dynamics_param ret is 0 175 {'position': [0.0, 0.0, 5.0], 'payload': 1.0, 'inertia': [0.0, 0.0, 0.0, 0.0, 0. 176 0, 0.0]} </pre>

65. set_none_tool_kinematics_param()

set_none_tool_kinematics_param()	
说明	设置无工具运动学参数
示例	<pre> 677 #设置无工具运动学参数 678 ret = robot.set_none_tool_kinematics_param() 679 print("set_none_tool_kinematics_param ret is {}".format(ret)) </pre>
输出	190 set_none_tool_kinematics_param ret is 0

66. set_tool_kinematics_param(tool_end_param)

set_tool_kinematics_param(tool_end_param)	
说明	设置工具的运动学参数 末端工具参数: tool_end_param={"pos": (x, y, z), "ori": (w, x, y, z)}
示例	<pre> 682 #设置工具运动学参数 683 tool_set_kin = {"pos":(0,0.2,0.1),"ori":(1,0,0,0)} 684 ret = robot.set_tool_kinematics_param(tool_set_kin) 685 print("set_tool_kinematics_param ret is {}".format(ret)) </pre>
输出	206 set_tool_kinematics_param ret is 0

67. get_tool_kinematics_param()

get_tool_kinematics_param()	
说明	获取工具的运动学参数 成功返回: 工具的运动学参数 tool_end_param={"pos": (x, y, z), "ori": (w, x, y, z)}
示例	<pre> 682 #设置工具运动学参数 683 tool_set_kin = {"pos":(0,0.2,0.1),"ori":(1,0,0,0)} 684 ret = robot.set_tool_kinematics_param(tool_set_kin) 685 print("set_tool_kinematics_param ret is {}".format(ret)) 686 #获取工具运动学参数 687 tool_get_kin = robot.get_tool_kinematics_param() 688 print("get_tool_kin_param = {}".format(tool_get_kin)) </pre>

输出	223 set_tool_kinematics_param ret is 0 224 get_tool_kin_param = {'pos': [0.0, 0.2, 0.1], 'ori': [1.0, 0.0, 0.0, 0.0]}
----	--

68. move_stop()

move_stop()	
说明	停止机械臂运动 需要在不同的线程中执行，还需要停止线程
示例	700 #停止机械臂运动 701 ret = robot.move_stop() 702 print("move_stop ret is {0}".format(ret))
输出	359 move_stop ret is 0

69. move_pause()

move_pause()	
说明	暂停机械臂运动 需要在不同的线程中执行
示例	690 #暂停机器人运动 691 ret = robot.move_pause() 692 print("move_pause ret is {0}".format(ret))
输出	371 move_pause ret is 0

70. move_continue()

move_continue()	
说明	继续机器人运动 需要在不同的线程中执行
示例	690 #暂停机器人运动 691 ret = robot.move_pause() 692 print("move_pause ret is {0}".format(ret)) 693 time.sleep(2) 694 #继续机器人运动 695 ret = robot.move_continue() 696 print("move_continue ret is {0}".format(ret))
输出	371 move_pause ret is 0 372 move_continue ret is 0

71. collision_recover()

collision_recover()	
说明	
示例	

输出	
----	--

72. get_robot_state()

get_robot_state()	
说明	<p>获取机械臂当前状态</p> <p>成功返回：机械臂当前状态</p> <ul style="list-style-type: none"> * 机械臂当前停止:RobotStatus.Stopped = 0 * 机械臂当前运行:RobotStatus.Running = 1 * 机械臂当前暂停:RobotStatus.Paused = 2 * 机械臂当前恢复:RobotStatus.Resumed = 3
示例	<p>机械臂在运行过程中</p> <pre> 711 #获取机器人运行状态 712 ret = robot.get_robot_state() 713 print("robot_state is {}".format(ret)) </pre>
输出	<pre> 467 robot_state is 1 </pre>

73. set_work_mode(mode)

set_work_mode(mode=0)	
说明	<p>设置机械臂服务器工作模式</p> <p>mode:服务器工作模式</p> <p>机械臂仿真模式:RobotRunningMode.RobotModeSimulator = 0</p> <p>机械臂真实模式:RobotRunningMode.RobotModeReal = 1</p>
示例	<p>设置为仿真模式</p> <pre> 715 #设置机械臂服务器工作模式 716 ret = robot.set_work_mode(0) 717 print("set_robot_work_mode ret is {}".format(ret)) </pre>
输出	<pre> 503 set_robot_work_mode ret is 0 </pre>

74. get_work_mode()

get_work_mode()	
说明	<p>获取机械臂服务器当前工作模式</p> <p>成功返回：服务器工作模式</p> <p>机械臂仿真模式:RobotRunningMode.RobotModeSimulator = 0</p> <p>机械臂真实模式:RobotRunningMode.RobotModeReal = 1</p>
示例	<pre> 715 #设置机械臂服务器工作模式 716 ret = robot.set_work_mode(0) 717 print("set_robot_work_mode ret is {}".format(ret)) 718 719 #获取机械臂服务器当前工作模式 720 ret = robot.get_work_mode() 721 print("get_robot_work_mode ret is {}".format(ret)) </pre>

输出	532	set_robot_work_mode ret is 0
	533	get_robot_work_mode ret is 0

75. set_collision_class(grade)

set_collision_class(grade=6)		
说明	设置机械臂碰撞等级，默认 6 grade 碰撞等级:碰撞等级 范围 (0~10) 需要连接真实机械臂，否则返回 nan	
示例	723	#设置碰撞等级
	724	ret = robot.set_collision_class(7)
	725	print("set_collision_class ret is {0}".format(ret))
输出	598	set_collision_class ret is 7

76. is_have_real_robot()

is_have_real_robot()		
说明	获取当前是否已经链接真实机械臂 成功返回：1: 存在 0: 不存在	
示例	727	#是否连接真实机械臂
	728	ret = robot.is_have_real_robot()
	729	print("is_have_real_robot ret is {0}".format(ret))
输出	连接了真实机械臂 610 is_have_real_robot ret is 1 未连接真实机械臂	

77. is_online_mode()

is_online_mode()		
说明	当前机械臂是否运行在联机模式 按下“MANUAL/LINKAGE”按钮，进入联机模式 成功返回：1: 在 0: 不在	
示例	731	#当前机械臂是否运行在联机模式
	732	ret = robot.is_online_mode()
	733	print("is_online_mode ret is {0}".format(ret))
输出	在联动模式 654 is_online_mode ret is 1 在手动模式 624 is_online_mode ret is 0	

78. is_online_master_mode()

is_online_master_mode()		
--------------------------------	--	--

说明	当前机械臂是否运行在联机主模式 按下按下“MANUAL/LINKAGE”按钮且“示教器使能”：联动主模式 按下按下“MANUAL/LINKAGE”按钮且“示教器不使能”：联动从模式 成功返回：1：主模式 0：从模式		
示例	735	#当前机械臂是否运行在联机主模式	
	736	ret = robot.is_online_master_mode()	
	737	print("is_online_master_mode ret is {0}".format(ret))	
输出	在联机主模式 668 is_online_master_mode ret is 1 在联机从模式 682 is_online_master_mode ret is 0 在手动模式 708 is_online_master_mode ret is 1		

79. get_joint_status()返回

get_joint_status()		
说明	获取机械臂当前状态信息 成功返回：返回六个关节状态，包括：电流，电压，温度 { 'joint1': { 'current': 电流(毫安), 'voltage': 电压(伏特), 'temperature': 温度(摄氏度) }, 'joint2': { 'current': 0, 'voltage': 0.0, 'temperature': 0 }, 'joint3': { 'current': 0, 'voltage': 0.0, 'temperature': 0 }, 'joint4': { 'current': 0, 'voltage': 0.0, 'temperature': 0 }, 'joint5': { 'current': 0, 'voltage': 0.0, 'temperature': 0 }, 'joint6': { 'current': 0, 'voltage': 0.0, 'temperature': 0 }}	
示例	739 740 741	#获取机械臂当前状态信息 ret = robot.get_joint_status() print("joint status = {0}".format(ret))
输出	750 joint status = { 'joint2': { 'current': -600, 'voltage': 48.04999923706055, 'tempera 751 ture': 28.700000762939453 }, 'joint3': { 'current': -292, 'voltage': 48.400001525878 752 906, 'temperature': 27.600000381469727 }, 'joint1': { 'current': -85, 'voltage': 48. 753 15999984741211, 'temperature': 30.299999237060547 }, 'joint6': { 'current': -173, 'v 754 oltage': 48.31999969482422, 'temperature': 37.400001525878906 }, 'joint4': { 'curren 755 t': 98, 'voltage': 47.9900016784668, 'temperature': 34.599998474121094 }, 'joint5': 756 { 'current': 0, 'voltage': 47.900001525878906, 'temperature': 35.70000076293945 }}	

80. get_current_waypoint()

get_current_waypoint()			
说明	获取机械臂当前位置信息 成功返回：关节位置信息		
示例	743	#获取机械臂当前位置信息	
	744	ret = robot.get_current_waypoint()	
	745	print("current waypoint is {0}".format(ret))	

输出	<pre> 770 current waypoint is {'joint': [0.0, -9.18522709980607e-06, -1.2246970072737895e-05 771 , -1.834014074120205e-05, 0.0, -7.336056114581879e-06], 'pos': [4.162393070034384e 772 -06, -0.2155, 0.9849999999690632], 'ori': [0.7071067811413445, 0.7071067811413445, 773 7.995417266726468e-06, -7.995417266726468e-06]} </pre>
----	---

81. get_board_io_config(io_type)

get_board_io_config(io_type=RobotIOType.User_D0)	
说明	<p>获取 IO 配置</p> <p>成功返回：IO 配置</p> <pre> * [{"id": ID * "name": "IO 名字" * "addr": IO 地址 * "type": IO 类型 * "value": IO 当前值},] </pre>
示例	<p>获取 User_D0 配置</p> <pre> 747 #获取板载IO配置 748 ioType = robotcontrol.RobotIOType.User_D0 749 ret = robot.get_board_io_config(ioType) 750 print("User_D0 Type is {0}".format(ret)) </pre>
输出	<pre> 782 User_D0 Type is [{'addr': 32, 'type': 5, 'id': 'U_D0_00', 'value': -1.0, 'name': ' 783 U_D0_00'}, {'addr': 33, 'type': 5, 'id': 'U_D0_01', 'value': -1.0, 'name': 'U_D0_0 784 1'}, {'addr': 34, 'type': 5, 'id': 'U_D0_02', 'value': -1.0, 'name': 'U_D0_02'}, { 785 'addr': 35, 'type': 5, 'id': 'U_D0_03', 'value': -1.0, 'name': 'U_D0_03'}, {'addr' 786 : 36, 'type': 5, 'id': 'U_D0_04', 'value': -1.0, 'name': 'U_D0_04'}, {'addr': 37, 787 'type': 5, 'id': 'U_D0_05', 'value': -1.0, 'name': 'U_D0_05'}, {'addr': 38, 'type' 788 : 5, 'id': 'U_D0_06', 'value': -1.0, 'name': 'U_D0_06'}, {'addr': 39, 'type': 5, ' 789 id': 'U_D0_07', 'value': -1.0, 'name': 'U_D0_07'}, {'addr': 40, 'type': 5, 'id': ' 790 U_D0_10', 'value': -1.0, 'name': 'U_D0_10'}, {'addr': 41, 'type': 5, 'id': 'U_D0_1 791 1', 'value': -1.0, 'name': 'U_D0_11'}, {'addr': 42, 'type': 5, 'id': 'U_D0_12', 'v 792 alue': -1.0, 'name': 'U_D0_12'}, {'addr': 43, 'type': 5, 'id': 'U_D0_13', 'value': 793 -1.0, 'name': 'U_D0_13'}, {'addr': 44, 'type': 5, 'id': 'U_D0_14', 'value': -1.0, 794 'name': 'U_D0_14'}, {'addr': 45, 'type': 5, 'id': 'U_D0_15', 'value': -1.0, 'name' 795 : 'U_D0_15'}, {'addr': 46, 'type': 5, 'id': 'U_D0_16', 'value': -1.0, 'name': 'U_D 796 0_16'}, {'addr': 47, 'type': 5, 'id': 'U_D0_17', 'value': -1.0, 'name': 'U_D0_17'} 797] </pre>

82. get_board_io_status(io_type, io_name)

get_board_io_status(io_type, io_name)	
说明	<p>获取 IO 状态</p> <p>io_type: 类型</p> <p>io_name: 名称 RobotUserIoName.user_dx_xx</p> <p>成功返回：IO 状态 double 数值(数字 IO, 返回 0 或 1, 模拟 IO 返回浮点数)</p>
示例	<p>获取 U_D0_00 的状态</p> <pre> 752 #获取板载IO状态 753 ioType = robotcontrol.RobotIOType.User_D0 754 ioName = robotcontrol.RobotUserIoName.user_do_00 755 ret = robot.get_board_io_status(ioType, ioName) 756 print("User_D0_00 status is {0}".format(ret)) </pre>
输出	U_D0_00 无输出

	809	User_DO_00 status is 0.0
	U_DO_00 有输出	
	821	User_DO_00 status is 1.0

83. set_board_io_status(io_type, io_name, io_value)

set_board_io_status(io_type, io_name, io_value)		
说明	设置 IO 状态 io_type: 类型 io_name: 名称 RobotUserIoName.user_dx_xx io_value: 状态数值(数字 IO, 0 (无效) 或 1 (有效); 模拟 IO: 浮点数) 成功返回: 0	
示例	<pre> 758 #设置板载IO状态,设置输出有效 759 ioType = robotcontrol.RobotIOType.User_DO 760 ioName = robotcontrol.RobotUserIoName.user_do_00 761 ioValue = 1 762 ret = robot.set_board_io_status(ioType,ioName,ioValue) 763 print("User_DO_00 status is {0}".format(ret)) </pre>	
输出	833	User_DO_00 status is 0

84. set_tool_power_type(power_type)

set_tool_power_type(power_type =RobotToolPowerType.OUT_0V)		
说明	设置工具端电源类型 power_type: 电源类型 RobotToolPowerType.OUT_0V RobotToolPowerType.OUT_12V RobotToolPowerType.OUT_24V	
示例	设置工具 IO 电源电压 24V <pre> 765 #设置工具端电源类型 766 toolIOPower = robotcontrol.RobotToolPowerType.OUT_24V 767 ret = robot.set_tool_power_type(toolIOPower) 768 print("set_tool_power_type ret is {0}".format(ret)) </pre>	
输出	845	set_tool_power_type ret is 0

85. get_tool_power_type()

get_tool_power_type()	
说明	获取工具端电源类型 成功返回: 电源类型, 包括如下: RobotToolPowerType.OUT_0V RobotToolPowerType.OUT_12V RobotToolPowerType.OUT_24V

示例	<pre> 765 #设置工具端电源类型 766 toolIOPower = robotcontrol.RobotToolPowerType.OUT_24V 767 ret = robot.set_tool_power_type(toolIOPower) 768 print("set_tool_power_type ret is {}".format(ret)) 769 770 #获取工具端电源类型 771 ret = robot.get_tool_power_type() 772 print("get_tool_power_type ret is {}".format(ret)) </pre>
输出	<pre> 845 set_tool_power_type ret is 0 846 get_tool_power_type ret is 2 </pre>

86. set_tool_io_type(io_addr, io_type)

**set_tool_io_type(io_addr=RobotToolIoAddr.TOOL_DIGITAL_IO_0,
io_type=RobotToolDigitalIoDir.IO_OUT)**

说明	<p>设置工具端数字 IO 类型</p> <p>io_addr:工具端 IO 地址 详见 class RobotToolIoAddr</p> <pre> class RobotToolIoAddr: TOOL_DIGITAL_IO_0 = 0 TOOL_DIGITAL_IO_1 = 1 TOOL_DIGITAL_IO_2 = 2 TOOL_DIGITAL_IO_3 = 3 def __init__(self): pass </pre> <p>io_type:工具端 IO 类型 详见 class RobotToolDigitalIoDir</p> <pre> class RobotToolDigitalIoDir: # 输入 IO_IN = 0 # 输出 IO_OUT = 1 def __init__(self): pass </pre> <p>成功返回: IO 类型, 包括如下: RobotToolDigitalIoDir.IO_IN RobotToolDigitalIoDir.IO_OUT</p>
示例	
输出	

87. get_tool_power_voltage()作用

get_tool_power_voltage()

说明	获取工具端电压数值
----	-----------

	成功返回：返回电压数值，单位（伏特）
示例	
输出	

88. get_tool_io_status(io_name)

get_tool_io_status(io_name)	
说明	获取工具端 IO 状态 io_name: IO 名称 成功返回：返回工具端 IO 状态
示例	<pre> 787 #获取tool DI_00状态 788 ioName = robotcontrol.RobotToolIoName.tool_io_0 789 ret = robot.get_tool_io_status(ioName) 790 print("Tool_DI_00 status is {0}".format(ret)) 791 792 #获取tool DO_01状态 793 ioName = robotcontrol.RobotToolIoName.tool_io_1 794 ret = robot.get_tool_io_status(ioName) 795 print("Tool_DO_01 status is {0}".format(ret)) </pre>
输出	T_DI_00 输入有效电平，T_DO_01 输出有效电平 <pre> 990 Tool_DI_00 status is 1.0 991 Tool_DO_01 status is 1.0 </pre>

89. set_tool_io_status(io_name, io_status)

set_tool_io_status(io_name, io_status)	
说明	设置工具端 IO 状态 io_name: 工具端 IO 名称 io_status: 工具端 IO 状态：取值范围（0 或 1）
示例	<pre> 800 #设置工具IO T_DO_01状态为有效 801 ioName = robotcontrol.RobotToolIoName.tool_io_1 802 ioStatus = 1 803 ret = robot.set_tool_io_status(ioName,ioStatus) 804 print("set T_DO_01 status ret is {0}".format(ret)) </pre>
输出	<pre> 992 set T_DO_01 status ret is 0 </pre>

90. startup_excit_traj_track(track_file, track_type, subtype)

startup_excit_traj_track(track_file='', track_type=0, subtype=0)	
说明	
示例	
输出	

91. `get_dynidentify_results()`

<code>get_dynidentify_results()</code>	
说明	
示例	
输出	

92. `set_robot_event_callback(callback)`

<code>set_robot_event_callback(callback)</code>	
说明	内部使用
示例	
输出	

第二篇 python linux32

一、 获取 python linux32 的 sdk 包

下载版本 V1.1.0 的 SDK。

Directory Listing For /dev/sdk/v1.1.0/linux/python/ - Up To /dev/sdk/v1.1.0/linux

Filename	Size	
libpyauboi5-v1.1.0.x64.tar.gz	10274.5 kb	We
libpyauboi5-v1.1.0.x86.tar.gz	16225.7 kb	We
readme.txt	0.7 kb	We



libpyauboi5-v1.1.0.
x86.tar.gz

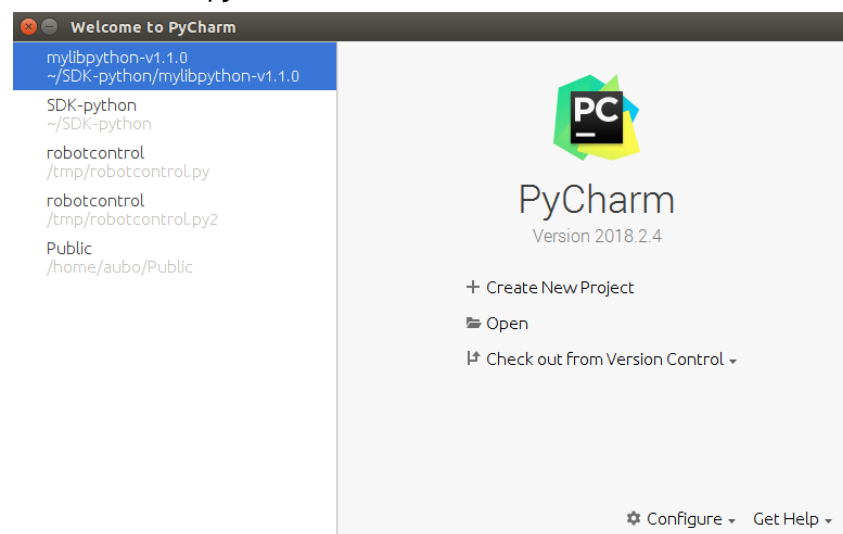
二、 linux 下 python 环境

1. 直接终端命令行方式

linux 默认安装了 python，可以直接命令行执行 python

2. python IDE 方式

推荐一款 IDE: pycharm。



三、 运行 python SDK

打开【libpyauboi5-v1.1.0.x86】->【robotcontrol.py】，修改 line 2114 行代码（机器人 ip 地址）。

ip = 'localhost': 表示本机 ip

```
2111     try:
2112
2113         # 链接服务器
2114         ip = 'localhost'
2115         port = 8899
2116         result = robot.connect(ip, port)
```

保存

回到终端

进入 robotcontrol.py 所在的路径

```
root@ubuntu:~/SDK-python/libpyauboi5-v1.1.0.x86# ls
config          libpyauboi5.so  libpyauboi5.so.1.1  logfiles
liblog4cplus-1.2.so.5  libpyauboi5.so.1  libpyauboi5.so.1.1.0  robotcontrol.py
```

输入命令: python robotcontrol.py, 回车

```
root@ubuntu:~/SDK-python/libpyauboi5-v1.1.0.x86# python robotcontrol.py
```

机械臂动作，终端打印出一些消息。