SDK Manual
[Python]

Document Version： V2.1.14
SDK Version： V2.1.14

Property Description

All rights reserved by JAKA Robotics Co., Ltd.

JAKA Robotics Co., Ltd owns the intellectual property rights of the related technologies contained in the products described in this document.

This document and related products are distributed under licenses that restrict their use, copy, distribution, and decompilation. Without the prior written authorization of JAKA Robotics Co., Ltd, no part of this product or this document may be copied in any way or in any form.

Version History

Version NO.	Version Date	Compatible Controller Version	Description
V1.0.0	2020.3.24	V1.4.10/V2.0.10	Create
V1.0.0	2020.6.24	V1.4.10/V2.0.10	Add additional instructions for different motion instructions
V1.0.14	2020.7.24	V1.4.10/V2.0.10	5.61 to 5.64 are new interfaces that modify jog_stop the interface
V1.0.15	2021.04.27	V1.4.24/V1.5.12.17/V2.0.24 and above	Add interfaces 4.69 to 4.110
V1.0.18	2021.08.30	V1.4.24/V1.5.12.17/V2.0.24 and above	Add API use instructions
V2.1.1	2021.12.10	V1.4.24/V1.5.12.17/V2.0.24 and above	Add FTP interface
V2.1.2	2022.07.01	V1.4.24/V1.5.12.17/V2.0.24 and above	Add part of interface and modify the manual structure
V2.1.3	2022.11.29	V1.4.24/V1.5.13.08/V2.0.24 add above	Designation the MoveC circles number
V2.1.7	2024.01.29	V1.5.13.08 and above	Add interfaces of getting and setting the robot installation angle Add interfaces return value -15 Fix the acceleration unit of MoveC Correct the description of upload_file Correct function name of set_status_data_update_time_interval
V2.1.8	2024.3.21	V1.5.13.08 and above	Add API usage instructions to get SDK log path
V2.1.11	2024.4.30	V1.5.13.08 and above	Fix: refine port 10004 communication Fix: fix set_ft_ctrl_frame interface not working Fix: fix for incorrect retransmission of port 10001 commands

V2.1.14	2024.09.30	V1.5.13.08 and above	Fix: 1. Redefined the port 10004 mechanism to solve the SDK crash issue previously caused by port 10004. 2. Redefined the motion block mechanism to solve the issue that the motion cannot be properly blocked. 3. Solved the inaccurate positioning issue of the inpos command. Add: 1. Added two new interfaces to set and get system variables: set_user_var(), get_user_var(). 2. Added one new interface to get the information on motion-related status: get_motion_status. 3. Add some embedded S related interfaces Header file: 1. Updated all the notes of header file to English. 2. Added copyright and version information at the beginning of header file.

Content

1 Introduction

This document is a python version of the Software Development Kit document.

2 Document Notes

- Running environment: linux python 3.5 32 bits,windows Python 3.7 64 bits
- Units used to parameters: mm, rad.
- In non-specific code examples, the robot is turned on by default and powered on
- All code samples in the documentation default to no interference in the robot's workspace
- The examples in the documentation are all default to the user's Python interpreter being able to find the jkrc module

2.1 Use under Linux

Linux needs to libjakaAPI.so and jkrc.so under the same folder and add the current folder path to the environment variable, export LD_LIBRARY_PATH=/xx/xx/

2.2 Use under Windows

Windows needs to put jkrc and jakaAPI .dll folder, and FAQs can be queried for frequently asked questions.

2.3 Dynamic library version number interrogate method

The normal use of jkrc requires a dynamic library file, and here's how to Interrogate the dynamic library version number:

Right-click the dll file in Windows, select the properties, and you can see the version information in the Details tab.

Enter the command strings in **Linux libjakaAPI.so | Grep jakaAPI_version**

3 Data Types

3.1 IO type

JAKA robots have three types of IO, which are control cabinet IO, tool IO, and extended IO, respectively.

The following IO definitions are used in the following sample code:

```
IO_CABINET              # Control cabinet IO
IO_TOOL                 # tool IO
IO_EXTEND               # extened IO
IO_REALY                # relay IO, DO is only supported on CAB V3 for now
IO_MOUDBUS_SLAVE        # Modbus slave IO, index from 0
IO_PROFINET_SLAVE       # Profinet slave IO, index from 0
IO_EIP_SLAVE            # Ethernet/IP slave IO, index from 0
```

3.2 Coordinate system type

JAKA robot has three coordinate systems, namely the base coordinate system / the current user coordinate system, joint space, and the tool coordinate system, respectively. The following coordinate system definitions are used in the following sample code:

```
COORD_BASE = 0          # the base coordinate system / the  current user coordinate system
COORD_JOINT = 1         # joint space
COORD_TOOL = 2          # the tool coordinate system
```

3.3 Sport type

There are two types of JAKA robot motion, which are defined as follows:

```
ABS = 0                 # Absolute move
INCR = 1                # Incremental move
CONTINUE = 2            # Continuous move
```

3.4 The return value values

Type	The return value	Describe
ERR_SUCC	0	Success

ERR_INVALID_HANDLER	-1	An invalid handle
ERR_INVALID_PARAMETER	-2	An invalid argument
ERR_COMMUNICATION_ERR	-3	There was a communication error
ERR_KINE_INVERSE_ERR	-4	The reverse solution failed
ERR_EMERGENCY_PRESSED	-5	The emergency stop key has not been released
ERR_NOT_POWERED	-6	The robot is not powered on
ERR_NOT_ENABLED	-7	The robot is not enabled
ERR_DISABLE_SERVOMODE	-8	SERVOMODE mode is not entered
ERR_NOT_OFF_ENABLE	-9	The power is not lowered before power is turned off
ERR_PROGRAM_IS_RUNNING	-10	The program is running
ERR_CANNOT_OPEN_FILE	-11	Opening the file failed
ERR_MOTION_ABNORMAL	-12	Abnormalities during the running process
ERR_FTP_PREFROM	-14	Abnormal FTP
ERR_VALUE_OVERSIZE	-15	Insufficient reserved memory
ERR_KINE_FORWARD	-16	Kine_forward error
ERR_EMPTY_FOLDER	-17	Not support empty folder
ERR_PROTECTIVE_STOP	-20	Protective stop
ERR_EMERGENCY_STOP	-21	Protective stop
ERR_SOFT_LIMIT	-22	On soft limit
ERR_CMD_ENCODE	-30	Fail to encode cmd string
ERR_CMD_DECODE	-31	Fail to decode cmd string
ERR_UNCOMPRESS	-32	Fail to uncompress port 10004 string
ERR_MOVEL	-40	Move linear error
ERR_MOVEJ	-41	Move joint error
ERR_MOVEC	-42	Move circular error
ERR_MOTION_TIMEOUT	-50	Block_wait timeout
ERR_POWERON_TIMEOUT	-51	Power on timeout
ERR_POWEROFF_TIMEOUT	-52	Power off timeout
ERR_ENABLE_TIMEOUT	-53	Enable timeoff
ERR_DISABLE_TIMEOUT	-54	Disable timeout
ERR_USERFRAME_SET_TIMEOU	-55	Set userframe timeout
ERR_TOOL_SET_TIMEOUT	-56	Set tool timeout
ERR_IO_SET_TIMEOUT	-60	Set io timeout

All The return value values of functions are a tuple except the function RC(ip)(as seen in 5.1 Instantiated robots). The return values of the Get class's functions are in the form of (**errcode, data**), the first element is the error code. When the robot is working well， the errcode is equal to 0. Otherwise, the errcode value is not equal to 0, and the value can be queried in the table above. The second element is data, such as joint angle values, and so on.

For example, the function ‘**get_joint_position()**’ is utilized to get the the joint angles. The return value is (0, [1 2 3 4 5 6]), which means that the joint angles is sucessfully obtained.

The return value is (-3,0), which indicates the joint angles cannot be obtained due to communication errors.

4 Interface

4.1 Basic Operation of Robots

4.1.1 Instantiated robots

Function	**RC(ip)**
Describe	Instantize a robot object
Parameters	ip: The robot's IP address needs to be filled in with a string only the correct IP address instantiated object to control the robot.
The return value	Success: Return a robot object Failed: The created object is destroyed

Sample Code：

```
import jkrc
robot = jkrc.RC("192.168.2.64")    # Instantizing a robot object
```

4.1.2 Login

Function	**login()**
Describe	Connect the robot controller
Parameters	
The return value	Success: (0,) Failed: Others

4.1.3 Logout

Function	**logout()**
Describe	Disconnect the controller
Parameters	

The return value	Success :(0,) Failed: Others

Sample Code：

```
import jkrc
robot = jkrc.RC("192.168.2.64")    # Instantizing a robot object
robot.login()                       #Connect the robot controller
pass
robot.logout()                      #Disconnect the controller
```

4.1.4 Power on

Function	**power_on()**
Describe	Turning on the robot and powering it on. The robot will have a delay of about 8 seconds
Parameters	
The return value	Success: (0,) Failed: Others

Sample Code：

```
import jkrc
robot = jkrc.RC("192.168.2.64")  # Instantizing a robot object
robot.login()          #Login
robot.power_on()    #Power on
robot.logout()        #logout
```

4.1.5 Power off

Function	**power_off()**
Describe	Turn off the robot
Parameters	
The return value	Success :(0,) Failed: Others

4.1.6 Shut down robot controller

Function	**shut_down()**
Describe	The robot control cabinet shuts down
Parameters	
The return value	Success :(0,) Failed: Others

4.1.7 Enable the robot

Function	**enable_robot()**
Describe	Enable the robot
Parameters	
The return value	Success :(0,) Failed: Others

Sample Code：

```
import jkrc
robot = jkrc.RC("192.168.2.64")  #Return a robot object
robot.login()                  #Login
robot.enable_robot()  #
robot.logout()            #Logout
```

4.1.8 Disable the robot

Function	**disable_robot()**
Describe	disable the robot
Parameters	
The return value	Success :(0,) Failed: Others

4.1.9 Get SDK Version No.

Function	**get_sdk_version()**
Describe	Get the SDK version number
Parameters	
The return value	Success: (0,version) Failed: Others

Sample Code：

```
''' Get SDK Version No.'''
    import jkrc
    robot = jkrc.RC("192.168.2.64")#
    robot.login()    #
    ret = robot.get_sdk_version()
    print("SDK version is:",ret[1])
```

```
robot.logout()     #Logout
```

4.1.10 Get the controller IP

Function	get_controller_ip ()
Describe	Get the controller IP
Parameter	
The return value	Success: (0, ip_list), ip_list: indicates the controller IP list. If the controller name is a specific value, the IP address of the controller corresponding to the controller name is returned. If the controller name is empty, all the IP address of the controllers in the network segment is returned Failure: Others

4.1.11 Enable drag mode

Function	drag_mode_enable(enable)
Description	Enable drag mode
Parameter	enable :TRUE means to enter the drag mode, FALSE means to quit the drag mode
The return value	Success: (0,) failed： other

Sample Code：

```
import jkrc
import time
#Coordiante System
COORD_BASE  = 0
COORD_JOINT = 1
COORD_TOOL  = 2
#motion mode
ABS = 0
INCR= 1
robot = jkrc.RC("192.168.2.160")
robot.login()
robot.power_on()
robot.enable_robot()
robot.drag_mode_enable(True)
ret = robot.is_in_drag_mode()
print(ret)
a = input()
robot.drag_mode_enable(False)
ret = robot.is_in_drag_mode()
print(ret)
robot.logout()
```

4.1.12 Interrogate whether in drag mode

Function	**is_in_drag_mode()**
Description	Interrogate whether in drag mode
Parameter	
The return value	Sucess：(0, state) state is equal to 1: the robot is in drag mode. state is equal to 0: the robot is not in drag mode。 Failed: Others

4.1.13 Set whether open SDK debug mode

Function	**set_debug_mode(mode)**
Description	Set whether the SDK enables debug mode.
Parameters	mode: Select TRUE to enter the debug mode. At this time, debugging information will be output in the standard output stream. When selecting FALSE,
Return value	Success: (0,) Failed: Others

4.1.14 Set SDK file path

Function	**set_SDK_filepath(filepath)**
Description	Set SDK file path
Parameters	filepath: File path
Return value	Success: (0,) Failed: Others

4.2 Rove Move

Planned motion with controller participation

4.2.1 Control robot movement in manual mode

Function	**Jog (aj_num , move_mode, coord_type, jog_vel, pos_cmd)**
Describe	Control robot movement in manual mode

Parameters	aj_num：Represent joint number [0-5] in joint space, and x, y, z, rx, ry, rz-axis in Cartesian space move_mode：Robot move mode, incremental move(0) or absolute move(1) 2 means continuous move coord_type：Robot move coordinate frame, tool coordinate frame, base coordinate frame (current world/user coordinate frame) or joint space jog_vel：Command velocity, unit of rotation axis or joint move is deg/s, move axis unit is mm/s pos_cmd：Command position, unit of rotation axis or joint move is rad, move axis unit is mm
The return value	Success :(0,) Failed: Others

Sample Code：

'''jog motion'''

'''1.joint space jog'''

```
# -*- coding: utf-8 -*-
import sys
sys.path.append('D:\\vs2019ws\PythonCtt\PythonCtt')
import time
import jkrc
PI=3.1415926
#coordinate
COORD_BASE  = 0
COORD_JOINT = 1
COORD_TOOL  = 2
#motion mode
ABS = 0
INCR= 1
#joint1-6,is correspond to 0-5 subscript

robot = jkrc.RC("192.168.2.194")#return robot
robot.login()#login
robot.power_on() #power on
robot.enable_robot()
print("move1")
robot.jog(0,INCR,COORD_JOINT,30*PI/180,PI/2)
time.sleep(5)#jog Is non-blocking instruction, and if jog instruction received in motion state will be discarded
print("move2")
robot.jog(0,INCR,COORD_JOINT,5,-90)
time.sleep(3)
robot.jog_stop()
robot.logout()
```

'''2.Cartesian space jog'''

```
import jkrc
import time
COORD_BASE = 0   # base coordinate
COORD_JOINT = 1  # joint
COORD_TOOL = 2   #tool coordinate
ABS = 0          # absolute motion
INCR = 1          # incremental motion
cart_x = 0       #x direction
cart_y = 1       #y direction
cart_z = 2       #z direction
cart_rx = 3       #rx direction
cart_ry = 4       #ry direction
cart_rz = 5       #rz direction
robot = jkrc.RC("192.168.2.64")#return a robot object
robot.login()    #login
robot.jog(aj_num = cart_z ,move_mode = INCR,coord_type = COORD_BASE ,jog_vel = 5,pos_cmd = 10)   # move 10mm in z+ direction
robot.jog_stop()
robot.logout()    #exit
```

Instruction:

If the robot is approaching a singular posture or joint limit, it will not be able to jog the robot using the above example code

4.2.2 Control robot movement stop in manual mode

Function	**jog_stop(joint_num)**
Describe	Stop the robot in manual mode
Parameters	joint_num: Robot axis number 0-5, when number is -1, stop all axes
The return value	Success :(0,) Failed: Others

Sample Code：

```
# -*- coding: utf-8 -*-
import sys
sys.path.append('D:\\vs2019ws\PythonCtt\PythonCtt')
import time
import jkrc
PI=3.1415926
#coordinate
COORD_BASE  = 0
COORD_JOINT = 1
COORD_TOOL  = 2
#motion mode
```

```
ABS = 0
INCR= 1
# Joints 1~6 correspond to 0~5 in order,

robot = jkrc.RC("192.168.2.160")#return a robot object
robot.login()#login
robot.power_on() #Power on
robot.enable_robot()
print("move1")
robot.jog(0,INCR,COORD_JOINT,30*PI/180,PI/2)
time.sleep(5)# jog is a non-blocking instruction, receiving jog instruction in motion state will be discarded
print("move2")
robot.jog(0,INCR,COORD_JOINT,5,-PI/2)
time.sleep(0.5)
robot.jog_stop(0)  # Stop after 0.5 seconds of movement, compared with the previous example code
robot.logout()
```

4.2.3 Robot joint motion

Function	**joint_move(joint_pos, move_mode ,is_block, speed)**
Description	Robot joint move
Parameter	joint_pos: Joint move position move_mode: 0 for absolute move， 1 for incremental move is_block： Set whether the interface is a block interface, TRUE represents a block interface and FALSE represents a non-block interface.Block means there will be no return value until robot completes its move, while non-block means there will be immediately a return value after the interface call is completed. speed: Robot joint move speed, unit: rad/s
The return value	Success: (0,) Failed: Others

Sample Code：
'''joint motion'''

```
# -*- coding: utf-8 -*-
# import sys
# sys.path.append('D:\\vs2019ws\PythonCtt\PythonCtt')
import time
import jkrc
PI=3.1415926
#motion mode
ABS = 0
INCR= 1
joint_pos=[PI/2,PI/3,0,PI/4,0,0]
robot = jkrc.RC("192.168.2.160")#return a robot object
```

```
robot.login()#login
robot.power_on() #上电
robot.enable_robot()
print("move1")
robot.joint_move(joint_pos,ABS,True,1)
time.sleep(3)
robot.logout()
```

4.2.4 Robot extension joint move

Function	joint_move_extend(joint_pos, move_mode, is_block, speed, acc, tol)
Description	Robot extension joint move. Add joint move angular acceleration and endpoint error.
Parameters	joint_pos: Angle of each target joint of robot joint move. move_mode: Specified move mode, 0 for absolute movement, 1 for incremental move, 2 for continuous move is_block: whether a block interface, True means block interface, False means non-block interface. speed: Robot joint move speed, unit: deg/s acc: Robot joint move angular acceleration. tol: Robot joint move endpoint error.
Return value	success: (0,) Failed: other

Sample Code：

```
import jkrc                    #import module
robot = jkrc.RC("192.168.2.226")  #returns a robot object
robot.login()
robot.joint_move_extend(joint_pos=[1, 1, 1, 1, 1, 1],move_mode=0, is_block=True,speed=20,acc=5, tol=0.1)
robot.joint_move_extend(joint_pos=[-1, 1, 1, 1, 1, 0],move_mode=0,is_block=true,speed=20,acc=5,  tol=0.1)
robot.logout() #exit
```

4.2.5 Robot end linear move

Function	linear_move(end_pos, move_mode, is_block ,speed)
Description	Robot end linear move
Parameter	end_pos: Robot end move target position move_mode: 0for absolute move，1for incremental move is_block: Set whether the interface is a block interface, TRUE represents a block interface and FALSE represents a non-block interface.Block means there will be no return value until robot completes its move, while non-block means there will be immediately a return value after the interface call is completed. speed: Robot linear move speed, unit: mm/s
Return value	Success: (0,) Failed: Others

Sample Code：
'''linear motion'''

```
import time
import jkrc
PI=3.1415926
#motion mode
ABS = 0
INCR= 1
tcp_pos=[0,0,-30,0,0,0]
robot = jkrc.RC("192.168.2.160")#return a robot object
robot.login()#login
robot.power_on() #Power on
robot.enable_robot()
print("move1")
# Blocking 30mm movement at 10mm/s in the negative direction of z-axis
ret=robot.linear_move(tcp_pos,INCR,True,10)
print(ret[0])
time.sleep(3)
robot.logout()
```

Note: Due to the difference on robot models, the calculation of kine inverse of the pose value filled in the example above could fail, thus Return value being -4.

4.2.6 Robot extension end linear move

Function	**linear_move_extend(end_pos, move_mode, is_block, speed, acc, tol)**
Description	Robot extension end linear move. Add space acceleration and space motion endpoint error
Parameters	end_pos: Robot end move target position. move_mode:Specify the movement mode,0 is absolute movement,1 is incremental movement is_block: whether a block interface, True means block interface, False means non-block interface. speed: Robot Cartesian space motion speed, unit: mm/s acc: Robot Cartesian space move acceleration, unit: mm/s^2. tol: Robot joint move endpoint error.
Return value	Success: (0,) Failed: Others

4.2.7 Arc movement at the robot end

Function	**circular_move (end_pos, mid_pos, move_mode, is_block, speed, acc, tol)**
Description	Arc movement at the robot end
Parameters	end_pos: Robot end move target position. mid_pos: Middle point of robot end move move_mode: Specified move mode, 0 for absolute movement, 1 for incremental move, 2 for continuous move is_block: whether a block interface, True means block interface, False means non-block interface.

	speed: Robot joint move speed, unit: mm/s acc: Robot joint move angular acceleration. tol: Robot joint move endpoint error.
Return value	Success: (0,) Failed: Others

Sample Code：

```
import jkrc                    #import module
robot = jkrc.RC("192.168.2.226")  #returns a robot object
robot.login()
robot.circular_move(end_pos = [100, 100, 0, 0, 1, 1],mid_pos = [100, 0, 0, 0, 1, 0],move_mode= 0, is_block = True, speed = 20, acc = 5, tol = 0.1)
robot.logout() #exit
```

Function	**circular_move_extend (end_pos, mid_pos, move_mode, is_block, speed, acc, tol, cricle_cnt, opt_cond)**
Description	Arc movement at the robot end
Parameters	end_pos: Arc move at the end of the robot. mid_pos: The middle point of robot end move. move_mode: Specify move mode, 0 is absolute move, 1 is incremental move, 2 is continuous move. is_block: Set whether the interface is a block interface, TRUE represents a block interface and FALSE represents a non-block interface. speed: Speed of robot MoveL(unit: mm/s) acc: Angular acceleration of robot MoveL(unit: mm/s2) tol: Robot motion endpoint error. circle_cnt: Circle number of MoveC opt_cond: Placeholder, enter none
Return value	Success: (0,) Failed: Others

Sample Code：

```
import jkrc
import math
import traceback


_ABS = 0
_BLOCK = 1


try:
    rc = jkrc.RC(“192.168.20.139”)
    rc.login()
    rc.power_on()
    rc.enable_robot()

    start_pos = [-406.250, 116.250, 374.000, math.pi, 0, math.pi * 0.5]
```

```
mid_pos = [-468.650, 211.450, 374.000, math.pi, 0, math.pi * 0.5]
end_pos = [-295.050, 267.450, 374.000, math.pi, 0, math.pi * 0.5]

rc.joint_move([0] + [math.pi * 0.5] * 3 + [math.pi * -0.5, 0], 0, 1, 200)
rc.linear_move(start_pos, _ABS, _BLOCK, 50)
rc.circular_move_extend(end_pos, mid_pos, _ABS, _BLOCK, 250, 1200, 5, None, 5)
except Exception:
traceback.print_exc()
```

4.2.8 Motion abort

Function	**motion_abort ()**
Description	Ability to terminate the robot's motion in any situation
Parameters	
Return value	Success: (0,) Failed: Others

Sample Code：

```
# -*- coding: utf-8 -*-
import sys
#sys.path.append('D:\\vs2019ws\PythonCtt\lib')
import time
import jkrc
robot = jkrc.RC("192.168.2.160")#Return a robot object
robot.login()#login
robot.power_on() #power on
robot.enable_robot()
print("move1")
robot.joint_move(joint_pos=[1,0,0,0,0,0],move_mode=1,is_block=False,speed=0.05)# Incremental motion
print("wait")
time.sleep(2)
print("stop")
robot.motion_abort()
robot.logout()#exit
```

4.2.9 Interrogate whether in position

Function	**is_in_pos()**
Description	Interrogate whether in position
Parameter	

Return value	Sucess：(0, state). state is equal to 1: The robot reaches the specified location state is equal to 0: The robot has not reached the specified location. Failed: Others

4.3 Set and Interrogate Robot Operation Info

4.3.1 Get robot status (simple)

Function	**Get_robot_status_simple()**
Description	Get the robot status
Parameter	
The return value	Success: (0, data) Errcode: error code Errmsg: error message Powered_on: whether the robot is powered on Enabled: whether the robot is enabled Failed: Others

4.3.2 Get robot status monitor data

Function	**get_robot_status ()**
Description	Get robot status data monitor data
Parameters	
Return value	success: (0, robotstatus), where the length of robotstatus data is 24, and the data are returned in an order as shown below: 1. errcode: the error code in case of a robot error, with 0 representing normal operation, and others representing abnormal operation 2. inpos: whether the robot is in position, 0 means robot still not move to position, 1 means robot has been moved to position 3. power_on: whether the robot is powered on, 0 means not powered on, 1 means powered on 4. enabled: whether the robot is enabled, 0 means not enabled, 1 means enabled 5. rapidrate: robot speed rate 6. protective_stop: whether the robot detects a collision, 0 means no collision detected, 1 means the opposite 7. drag_status: whether the robot is in drag status, 0 means not in drag status, 1 means the opposite 8. on_soft_limit: whether the robot is on limit, 0 means not triggered limit protection, 1 means triggered limit protection 9. current_user_id: the current user coordinate frame ID used by the robot 10. current_tool_ID: the current tool coordinate frame ID used by the robot 11. dout: robot control cabinet digital output signal 12. din: robot control cabinet digital input signal

	13. aout: robot control cabinet analog output signal 14. ain: robot control cabinet analog input signal 15. tio_dout: robot end tool digital output signal 16. tio_din: robot end tool digital input signal 17. tio_ain: robot end tool analog input signal 18. extio: robot external extension IO module signal 19. cart_position: robot end Cartesian space position value 20. joint_position: robot joint space position 21. robot_monitor_data: robot status monitor data (scb major version, scb minor version, controller temperature, robot average voltage, robot average current, monitor data of 6 robot joints (transient current, voltage and temperature)) 22. torq_sensor_monitor_data: robot torque sensor status monitor data (torque sensor IP addresses and port numbers, tool payload (payload mass, centroid x-axis, y-axis and z-axis coordinates), torque sensor status and exception error code, actual contact force values of 6 torque sensors and original reading values of 6 torque sensors) 23. is_socket_connect: whether the connection channel of SDK and controller is normal, 0 means abnormal connection channel, 1 means normal connection channel 24. emergency_stop whether the robot is emergency stopped, 0 means not pressed the emergency-stop button, 1 means the opposite 25. tio_key Robot end tool button [0] free; [1] point; [2] end light button Failed: Others **Notes:** **If the robot has no corresponding IO, it will return such a string as "no extio".** The error code file can be found in the **JAKA secondary development kit \ reference material**

Sample Code：

```
import sys
import jkrc
import time

robot = jkrc.RC("192.168.2.160")
robot.login()
ret = robot.get_robot_status()
if ret[0] == 0:
    print("the joint position is :",ret[1])
    print(len(ret[1]))
    for i in range(len(ret[1])):
        print(ret[1][i])
else:
    print("some things happend,the errcode is: ",ret[0])
robot.logout()
```

4.3.3 Set robot status data update time interval

Function	**Set_status_data_update_time_interval (millisecond)**
Description	Set the robot status data update time interval to change the system occupied storage, the default is to send as quick as possible to ensure the data is the newest (4ms)
Parameter	Millisecond: time parameters, uint: ms.

The return value	Success: (0,) Failed: Others

Sample Code:

```
import jkrc                          #import modules
robot = jkrc.RC("192.168.2.165")  #return to a robot
robot.login()
robot.set_status_data_update_time_interval(100)
robot.logout()
```

4.3.4 Get joint angles

Function	**get_joint_position()**
Description	Get the robot 6 joint values
Parameter	
The return value	Success: joint_pos is a tuple with 6 elements (j1, j2, j3, j4, j5, j6), J1, J2, J3, J4, J5, and j6 represent the Angle values from joint 1 to joint 6 respectively Failed: Others

Sample Code：

'''get joints degrees'''

```
# -*- coding: utf-8 -*-
import sys
#sys.path.append('D:\\vs2019ws\PythonCtt\lib')
import time
import jkrc
PI=3.1415926
robot = jkrc.RC("192.168.2.160")#returns a robot object
ret = robot.login()#login
ret = robot.get_joint_position()
if ret[0] == 0:
    print("the joint position is :",ret[1])
else:
    print("some things happend,the errcode is: ",ret[0])
robot.logout()  #exit
```

4.3.5 Get TCP pose

Function	**get_tcp_position()**
Description	Get tcp pose
Parameter	

Return value	Success: (0,Cartesian_pose), where Cartesian_pose is a tuple containing 6 elements (x,y,z,rx,ry,rz), with x,y,z,rx,ry,rz representing the pose value of robot tool end Failed: Others

Sample Code:

''' Get TCP pose '''

```
# -*- coding: utf-8 -*-
import sys
#sys.path.append('D:\\vs2019ws\PythonCtt\lib')
import time
import jkrc
PI=3.1415926
robot = jkrc.RC("192.168.2.160")#returns a robot object
ret = robot.login()#login
ret = robot.get_tcp_position()
if ret[0] == 0:
    print("the tcp position is :",ret[1])
else:
    print("some things happend,the errcode is: ",ret[0])
robot.logout()  #exit
```

4.3.6 Set user coordinate frame parameter

Function	**set_user_frame_data(id, user_frame, name)**
Description	Set the parameter of specified user coordinate frame
Parameter	id :The value range of the user coordinate frame number is [1,10]， 0 for robot-based coordinate frame user_frame: Offset value of user coordinate frame[x,y,z,rx,ry,rz] name:Alias of user coordinate frame
Return value	Success: (0,) Failed: Others

4.3.7 Get user coordinate frame data

Function	**get_user_frame_data()**
Description	Get user coordinate frame ID
Parameters	
Return value	Success: (0, id, tcp) id The value range of the user coordinate frame number is [1,10], where 0 represents robot basic coordinate frame tcp: Offset value of user coordinate frame[x,y,z,rx,ry,rz] Failed: Others

4.3.8 Get user coordinate frame ID

Function	**get_user_frame_id(id)**

Description	Get user coordinate frame ID
Parameter	
Return value	Success: (0, id) The value range of the id is [0,10], where 0 represents the world coordinate frame Failed: Others

4.3.9 Set user coordinate frame ID

Function	**set_user_frame_id(id)**
Description	Set user coordinate frame ID
Parameter	id: The value range of the user coordinate frame ID is [0,10], where 0 represents the world coordinate frame
Return value	Success: (0,) Failed: Others

4.3.10 Set the tool information for the specified number

Function	**set_tool_data(id, tcp, name)**
Description	Set the tool information for the specified number
Parameters	id : Set the tool ID, optional ID is 1 to 10, 0 means end flange, already used by controller. tcp : Set tool coordinate system parameters [x,y,z,rx,ry,rz]. name: alias of user coordinate system
Return values	Success: (0,) Failed: Others

4.3.11 Set tool ID

Function	**set_tool_id(id)**
Description	Set tool ID
Parameter	Id:The value range of tool coordinate frame ID is [0,10], 0 means no tools, flange center
Return value	Success: (0,) Failed: Others

4.3.12 Interrogate target tool coordinate frame ID

Function	**get_tool_data(id)**
Description	Interrogate the tool ID used by robot
Parameters	
Return value	Success: (0,(id,tcp)), where the ID values range from 0 to 10, with 0 representing the end flange, which has been used by controller.

	tcp: tool coordinate frame parameter [x,y,z,rx,ry,rz] Failed: Others

4.3.13 Get the tool ID currently in use

Function	**get_tool_id()**
Description	Get the tool ID currently in use
Parameter	
Return value	Success: (0, id), where the ID values range from 0 to 10, with 0 representing the end flange, which has been used by controller. Failed: Others

Sample Code:

```python
# -*- coding: utf-8 -*-
import sys
sys.path.append('D:\\vs2019ws\PythonCtt\PythonCtt')
import time
import jkrc
PI=3.1415926

robot = jkrc.RC("192.168.2.160") #returns a robot object
robot.login()                      #login
ret = robot.get_tool_data(1)    #Get tool coordinate system data
if ret[0] == 0:
    print("the tool data is :",ret)
else:
    print("some things happend,the errcode is: ",ret[0])
robot.set_tool_data(1,[0,0,1,0,0,0],'testlx') # Set tool coordinate system data
time.sleep(0.5)
ret = robot.get_tool_data(1)    #Get tool coordinate system data
if ret[0] == 0:
    print("the tool data is :",ret)
else:
    print("some things happend,the errcode is: ",ret[0])
ret = robot.get_tool_id()     #Get tool coordinate system id
print("tool_id",ret)
robot.set_tool_id(1)         #Set tool coordinate system data
time.sleep(0.5)
ret = robot.get_tool_id()     # Get tool coordinate system id
print("tool_id",ret)
robot.logout()
```

4.3.14 Set digital output variables

Function	**set_digital_output(iotype = a_type, index = a_number, value = a_value)**
Description	Set DO Value

Parameter	Iotype: DO Type Index: DO Index Value: DO Value
Return value	Success: (0,) Fail: other

Sample Code：

''' Set the value of DO3 to 1 ‘''

```
# -*- coding: utf-8 -*-
import sys
sys.path.append('D:\\vs2019ws\PythonCtt\PythonCtt')
import time
import jkrc
PI=3.1415926

IO_CABINET  = 0  #controller panel IO
IO_TOOL = 1     #Tool IO
IO_EXTEND = 2   #extension IO

robot = jkrc.RC("192.168.2.160")
robot.login()
ret = robot.get_digital_output(0,2)
if ret[0] == 0:
    print("1the DO2 is :",ret[1])
else:
    print("some things happend,the errcode is: ",ret[0])
robot.set_digital_output(IO_CABINET, 2, 1)# Set the pin output value of DO2 to 1
time.sleep(0.1)
ret = robot.get_digital_output(0, 2)
if ret[0] == 0:
    print("2the DO2 is :",ret[1])
else:
    print("some things happend,the errcode is: ",ret[0])
robot.logout()  #exit
```

4.3.15 Set analog output variables

Function	**set_analog_output(iotype = a_type, index = a_number, value = a_value)**
Description	Set analog output (AO) value
Parameter	Iotype: AO Type Index: AO Index Value: AO Settings
Return value	Success: (0,) Failed: Others

Sample Code：

'''Set the value of AO4 to 1.55'''

```
Example：
''' Set the value of AO4 to 1.55'''
import jkrc
IO_CABINET  =0  #controller panel IO
IO_TOOL = 1      #Tool IO
IO_EXTEND = 2   #extension IO
robot = jkrc.RC("192.168.2.64")
robot.login()
robot.set_analog_output(iotype = IO_CABINET,index = 3,value = 1.55)#
robot.logout()  #exit
```

4.3.16 Get digital input status

Function	**get_digital_input(iotype, index)**
Description	Interrogate DI status
Parameter	Iotype：DI Type Index：DI Index (starting from 0)
Return value	Succes：(0,value)，value: Interrogate result of DI status Failed: Others

4.3.17 Get digital output status

Function	**get_digital_output(iotype, index)**
Description	Interrogate DO status
Parameter	Iotype：DO Type Index：DO Index
Return value	Success: (0,value) value:Interrogate result of DO status Failed: Others

4.3.18 Get analog input variables

Function	**get_analog_input(iotype, index)**
Description	Get the type of AI value
Parameter	iotype: AI Type index: AI Index (starting from 0)
Return value	Success: (0,value) result：Interrogate result of AI status (expressed as a floating point), Failed: Others

4.3.19 Get analog output variables

Function	**get_analog_output(type, index)**
Description	Get AO value
Parameter	type: AO Type index: AO Index (starting from 0)
Return value	Success: (0,value) result：Interrogate result of AO status (expressed as a floating point), Failed: Others

Sample Code：

'''Interrogate the value of AO4 to'''

```
Example：
'''Interrogate the value of AO4 to '''
import jkrc
IO_CABINET  =0  # controller panel IO
IO_TOOL = 1     #Tool IO
IO_EXTEND = 2   #Extension IO
robot = jkrc.RC("192.168.2.64")
robot.login()
robot.set_analog_output(iotype = IO_CABINET,index = 3,value = 1.55)
ret = robot.get_analog_output(iotype = IO_CABINET,index = 3)
print("AO value is:",ret[1])
robot.logout()  #exit
```

4.3.20 Interrogate whether extension IO in running status

Function	**is_extio_running()**
Description	Interrogate whether the extension IO module is running
Parameter	
Return value	Success: (0,status) That status is 1 means running Failed: Others

4.3.21 Set payload

Function	**set_payload(mass = m, centroid = [x,y,z])**
Description	Set payload
Parameter	mass: payload mass, unit: kg centroid: payload centroid coordinates[x, y, z], unit: mm

Return value	Success: (0) Failed: Others

Example：

```
import jkrc                          #import module
robot = jkrc.RC("192.168.2.226")     #returns a robot object
robot.login()
robot.set_payload(mass= 1, centroid =[0.01,0.02,0.03])
robot.logout()                                          #exit
```

4.3.22 Get payload data

Function	**get_payload()**
Description	Get payload data
Parameter	
Return value	Success: (0 , payload) The expression of payload is (mass, (x, y, z)). payload is a tuple,whose length is 2.The first element of tuple is the mass of payload.The second element of tuple is the centroid of payload. Failed: Others

```
"""  Set payloads"'
import jkrc
robot = jkrc.RC("192.168.2.64") # returns a robot object
robot.login()                    #exit
robot.set_payload(mass= 1,centroid =[0.01,0.02,0.03])
ret = robot.get_payload()
if ret[0] == 0:
    print("the payload is :",ret[1])
else:
    print("some things happend,the errcode is: ",ret[0])
robot.logout()
```

4.3.23 Set tioV3 parameters

Function	**set_tio_vout_param（vout_enable ,vout_vol）**
Description	Set tioV3 parameters
Parameters	vout_enable voltage enable，0:close， 1 open vout_vol voltage volume 0:24v 1:12v
Return value	Success: (0) Failed: Others

4.3.24 Get tioV3 parameters

Function	**get_tio_vout_param（vout_enable ,vout_vol）**

Description	Get tioV3 parameters
Parameters	vout_enable voltage enable，0:close，1 open vout_vol voltage volume 0:24v 1:12v
Return value	Success: (0,(vout_enable ,vout_vol)) Failed: Others

4.3.25 Get robot state

Function	**get_robot_state()**
Description	Get robot state
Parameter	
Return value	Sucess: (0, state), where state is a tuple containing 3 elements, representing whether emergency stop, whether power on, and whether servo enable, respectively. 1 for yes, 0 for no. Failed: Others

Sample Code：

```
# -*- coding: utf-8 -*-
import sys
#sys.path.append('D:\\vs2019ws\PythonCtt\lib')
import time
import jkrc

robot = jkrc.RC("192.168.2.160")#returns a robot object
ret = robot.login()#login

ret = robot.get_robot_state()
if ret[0] == 0:
    print("the robot state is :",ret[1])
else:
    print("some things happend,the errcode is: ",ret[0])
robot.logout()  #exit
```

4.3.26 TIO add or modify semaphore

Function	**add_tio_rs_signal（sign_info）**
Description	Add or modify semaphore
Parameters	sign_info: dict Semaphore attribute
Return value	Success: (0,) Failed: Others

Sample Code:

```
def example_add_tio_rs_signal():
        rc = jkrc.RC(_RC_ADDRESS)
        rc.login()
        ret = rc.add_tio_rs_signal({
```

```
        'sig_name': 'signal_tmp',  //Signal name
        'chn_id': 0,  //RS485channel ID
        'sig_type': 0,           //Semaphore type
        'sig_addr': 0x1,  //Register address
        'value': 5,           //Value Invalid when setting
        'frequency': 5  // The refresh frequency of semaphore in the controller is not more than 10.
    })
```

4.3.27 TIO delete semaphore

Function	**del_tio_rs_signal (sign_name)**
Description	Add or modify semaphore
Parameters	sign_info: str Semaphore identification name
Return value	Success: (0,) Failed: Others

Sample Code:

```
def example_del_tio_rs_signal():
    rc = jkrc.RC(_RC_ADDRESS)
    rc.login()
    ret = rc.del_tio_rs_signal('signal_tmp')
    print('ret is {}'.format(ret))
```

4.3.28 TIO RS485 send command

Function	**send_tio_rs_command(chn_id, cmd)**
Description	RS485 send command
Parameters	chn_id: int channel number data: str Data bit ; Convert hex string to byte array
Return value	Success: (0,) Failed: Others

Sample Code:

```
def    example_send_tio_rs_command():
       rc = jkrc.RC(_RC_ADDRESS)
       rc.login()
       ret = rc.send_tio_rs_command( 2,
       bytearray.fromhex(“01 06 01 00 00 01”))
       print('ret is {}'.format(ret))
```

4.3.29 TIO get semaphore information

Function	**get_rs485_signal_info()**
Description	RS485 semaphore information
Parameters	
Return value	Success: (0,sign_info_list) sign_info_list: list Semaphore information array Failed: Others

Sample Code:

```
def example_get_rs485_signal_info():
        rc = jkrc.RC(_RC_ADDRESS)
        rc.login()
        ret, sign_info_list = rc.get_rs485_signal_info()
        print('ret is: {}, sign_info_list: {}'.format(ret, sign_info_list))
        # [{'value': 0, 'chn_id': 0, 'sig_addr': 0, 'sig_name': '', 'sig_type': 0, 'frequency'
    0}, ...] -> 4.3.26TIOadd or modify semaphore
```

4.3.30 TIO Setting TIO mode

Function	**set_tio_pin_mode(pin_type, pin_mode)**
Description	Set tio mode
Parameters	pin_type: tio type 0 for DI Pins, 1 for DO Pins, 2 for AI Pins pin_mode: tio mode **DI Pins**: 0:0x00 DI2 is NPN, DI1 is NPN, 1:0x01 DI2 is NPN, DI1: PNP, 2:0x10 DI2 is PNP, DI1 is NPN, 3:0x11 DI2 is PNP, DI1 is PNP **DO Pins:** The lower 8 bits of data and the upper 4 bits are configured as DO2, The lower four bits are DO1 configuration, 0x0 DO is NPN output, 0x1 DO PNP output, 0x2 DO is push-pull output, 0xF RS485H interface **AI Pins**: 0: Analog input function is enabled, RS485L is disabled, 1: RS485L interface is enabled and analog input function is disabled.

Return value	Success: (0,) Failed: Others

4.3.31 TIO get TIO mode

Function	**get_tio_pin_mode(pin_type)**
Description	Get tio mode
Parameters	pin_type: tio type 0 for DI Pins, 1 for DO Pins, 2 for AI Pins
Return value	Success: (0, pin_mode) Failed: Others

4.3.32 TIO RS485 communication parameter configuration

Function	**set_rs485_chn_comm(dict)**
Description	Configure RS485 communication parameters
Parameters	When the channel mode is set to Modbus RTU, specify the Modbus slave ID Additionally The parameter type is a dictionary: dict = { 'chn_id':1, 'slave_id':1, 'baudrate': 115200, 'databit': 8, 'stopbit': 1, 'parity': 78 }
Return value	Success: (0,) Failed: Others

4.3.33 TIO RS485 Communication Parameter Query

Function	**get_rs485_chn_comm(chn_id, slave_id, baudrate, databit,stopbit, parity)**
Description	Query RS485 communication parameters
Parameters	mod_rtu_com When querying, chn_id serves as an input parameter
Return value	Success: (0, (chn_id, slave_id,baudrate, databit, stopbit, parity)) Failed: Others

4.3.34 TIO RS485 communication mode configuration

Function	**set_rs485_chn_mode(chn_id, chn_mode)**
Description	Configure RS485 communication mode
Parameters	chn_id 0: RS485H, channel 1; 1: RS485L, channel 2 chn_mode 0: Modbus RTU, 1: Raw RS485, 2: Torque Sensor
Return value	Success: (0, (chn_id, slave_id,baudrate, databit, stopbit, parity)) Failed: Others

4.3.35 TIO RS485 communication mode configuration

Function	**get_rs485_chn_mode(chn_id)**
Description	Query RS485 communication mode
Parameters	chn_id Input parameter: 0 for RS485H, channel 1; 1 for RS485L, channel 2
Return value	Success: (0, chn_mode) chn_mode 0: Modbus RTU, 1: Raw RS485, 2: Torque Sensor Failed: Others

4.3.36 Set robot installation angle

Function	**set_installation_angle(anglex, angley)**
Description	Set the installation angle
Parameters	anglex: Rotation angle around the X-axis, range: 0-180 deg anglez: Rotation angle around the Z-axis, range: 0-360 deg
Return value	Success: (0,) Failed: Others

Sample Code:

```python
import jkrc
rc = jkrc.RC("192.168.137.152")
res = rc.login()
if res[0] != 0:
    raise "rc login failed."

anglex = 180
angley = 0
res = rc.set_installation_angle(anglex, angley)
if res[0] != 0:
    raise "set installation angle failed."

res = rc.get_installation_angle()
if res[0] != 0:
    raise "get installation angle failed."

print("installation angle:")
print("quat: [{x}, {y}, {z}, {s}]".format(s=res[1][0], x=res[1][1], y=res[1][2], z=res[1][3]))
print("rpy: [{rx}, {ry}, {rz}]".format(rx=res[1][4], ry=res[1][5], rz=res[1][6]))
rc.logout()

```

4.3.37 Get robot installation angle

Function	get_installation_angle()
Description	Get the installation angle of robot
Parameters	
Return value	Success：(0, [qs, qx, qy, qz, rx, ry, rz]) qs, qx, qy, qz: Quaternion representing the installation angle rx, ry, rz: Installation angles in Roll-Pitch-Yaw (RPY) format Failed: Others

4.4 Robot Safety Status Settings

4.4.1 Interrogate whether on limit

Function	is_on_limit()
Description	Interrogate whether on limit
Parameter	
Return value	Sucess： (0, state). state is equal to 1: The robot movement is beyond the limited range state is equal to 0: The robot movement is in the limited range Failed: Others

4.4.2 Interrogate whether in Collision Protection mode

Function	is_in_collision()
Description	Interrogate whether in Collision Protection mode
Parameter	
Return value	Sucess： (0, state). state is equal to 1: the Collision Protection mode is active. state is equal to 0: the Collision Protection mode is not active. Failed: Other

4.4.3 Clear the robot error code after collision occur

Function	collision_recover()
Description	Clear the robot error code after collision occur
Parameters	
Return value	Success: (0,) Failed: Other

Sample Cide：

```
from typing import Counter
import jkrc
import time


robot = jkrc.RC("192.168.2.160")
robot.login()
robot.power_on()
robot.enable_robot()
ret = robot.get_collision_level()#gets the current collision level
print(ret)
robot.set_collision_level(1)#set collision level
ret = robot.get_collision_level()
print(ret)
num = 0
while(1):
    ret = robot.is_in_collision()  #Interrogate whether in Collision Protection mode
    collision_status = ret[1]
    if collision_status == 1:
        time.sleep(5)
        robot.collision_recover()   #Clear the robot error code after collision occur
        print(" in collision "+ str(num))
    else:
        print("the robot is not in collision "+ str(num))
    time.sleep(1)
    num=num+1

robot.logout()
```

4.4.4 Set collision level

Function	**set_collision_level(level)**
Description	Set collision level
Parameter	level: the range of collision value is [0,5], 0: close collision, 1: collision threshold 25N, 2: collision threshold 50N, 3: collision threshold 75N, 4: collision threshold 100N, 5: collision threshold 125N
Return value	Success: (0,) Failed: Others

4.4.5 Get collision level

Function	**get_collision_level()**
Description	get collision level
Parameter	
Return value	Success: (0,level) level: the collision level. 0: close collision, 1: collision threshold 25N, 2: collision threshold 50N, 3: collision threshold 75N, 4: collision threshold 100N, 5: collision threshold 125N

4.4.6 Get the last error code

Function	**get_last_error ()**
Description	Get the last error code in the robot running process, when clear_error is called, the last error code will be cleared. If you need to use the get_last_error interface, set the error code file pathIf you just need to get the error code, there is no need to call set_errorcode_file_path.
Parameter	
Return value	Success: (0, error) Failed: Others

Sample Code：

```python
import jkrc
robot = jkrc.RC("192.168.2.194")#return a robot object
robot.login()  #login
robot.program_load("not_exist") #Intentionally load a non-existent program, causing an error
ret = robot.get_last_error ()#Without setting the error code file path, you can only get the error code, not the specific error information
print(ret[1])
robot.set_errorcode_file_path("D:\\JAKA_ERROR_CODE.csv") # The error file path cannot contain Chinese
ret = robot.get_last_error ()  # You can get the error code and specific error information after setting error code file path
print(ret[1])
robot.logout()  #exit
```

4.4.7 Set error code file path

Function	set_errorcode_file_path (errcode_file_path)
Description	Set the error code file path. If you need to use the get_last_error interface, set the error code file path. If no need to use the get_last_error interface, do not set the interface. **Note: The path can not contain any Chinese characters, otherwise it will not work**
Parameter	errcode_file_path： The path where the error code file is stored. The error code file can be found in the JAKA secondary development kit\reference material
Return value	Success: (0,) Failed: Others

4.4.8 Clear error status

Function	clear_error()
Description	Clear error status.
Parameters	
Return value	Success: (0,) Failed: Others

Sample Code：

Refer to collision protection mode recovery

4.4.9 Set the robot automatic move termination types upon network exceptions

Function	set_network_exception_handle (millisecond, mnt)
Description	Set the network exception control handle to control the robot motion status upon network exceptions.
Parameters	millisecond: The time parameter, unit: ms. mnt: the motion types the robot needs to perform upon network exceptions. 0 means that the robot should keep its current move, 1 means that the robot should pause its move and 2 means the robot should stop its move.
Return value	Success: (0,) Failed: Others

Sample Code：

```
# -*- coding: utf-8 -*-
# import sys
# sys.path.append('D:\\vs2019ws\PythonCtt\PythonCtt')
import time
import jkrc
PI=3.1415926
#motion mode
ABS = 0
```

```
INCR= 1
robot = jkrc.RC("192.168.2.160")#return a robot object
robot.login()#login
robot.power_on() #power on
robot.enable_robot()
robot.set_network_exception_handle(100,2)#Set 100ms, move abort.
print("move1")
num=0
while(1):
    robot.joint_move([1,1,1,1,1,1],ABS,False,0.5)
    robot.joint_move([-1,1,1,1,1,1],ABS,False,0.5)
    num = num +1
    print(num)
    time.sleep(6)
robot.logout()
```

4.5 Use APP Script Program

4.5.1 Load the specified program

Function	**program_load(file_name)**
Description	Run the loaded program
Parameter	file_name :program name likes "file_name"
Return value	Success: (0,) Failed: Others

4.5.2 Get the loaded program

Function	**get_loaded_program()**
Description	Get the name of the loaded operating program
Parameter	
Return value	Success: (0,file_name) Failed: Others

Sample Code：

```
import jkrc
robot = jkrc.RC("192.168.2.64")#returns a robot object
robot.login()  #login
ret = robot.program_load("program_test")# Load the script program_test written through the app, you need to write it yourself
ret = robot.get_loaded_program()
print("the loaded program is:",ret[1])
robot.logout() #exit
```

4.5.3 Get current line

Function	**get_current_line()**
Description	Get current line
Parameters	
Return value	Success: (0,curr_line),curr_line: Interrogate result of current line number. Failed: Others

4.5.4 Run the loaded program

Function	**program_run()**
Description	Run the loaded program
Parameter	
Return value	Success: (0,) Failed: Others

4.5.5 Pause the running program

Function	**program_pause()**
Description	Pause the running program
Parameter	
Return value	Success: (0,) Failed: Others

4.5.6 Resume program

Function	**program_resume()**
Description	Resume program
Parameter	
Return value	Success: (0,) Failed: Others

4.5.7 Abort program

Function	**program_abort()**

Description	Abort program
Parameter	
Return value	Success: (0,) Failed: Others

4.5.8 Get program status

Function	**get_program_state()**
Description	Get the program status
Parameter	
Return value	Success: (0, state) That state is equal to 0 stands for stoped. That state is equal to 1 stands for running. That state is equal to 2 stands for paused. Failed: Others

Sample Code:

```
# -*- coding: utf-8 -*-
import sys
sys.path.append('D:\\vs2019ws\PythonCtt\PythonCtt')
import time
import jkrc
import _thread
PI=3.1415926

def print_state(name,robot):
    while(1):
        ret = robot.get_program_state() # Interrogate the program running status, 0 represents for Program terminated or no program running, 1 represents for Program running, 2 represents for Pause
        print("the robot program state is:",ret[1])
        time.sleep(1)

robot = jkrc.RC("192.168.2.160")#returns a robot object
robot.login()  #login
ret = robot.program_load("simple")# Load the script program_test written through the app, you need to write it yourself
ret = robot.get_loaded_program()
print("the loaded program is:",ret[1])
robot.program_run()
_thread.start_new_thread( print_state,("p1_state", robot))# Open a "p1" thread to interrogate the program status
time.sleep(10)
robot.program_pause() #pause
time.sleep(10)
robot.program_resume() #resume
time.sleep(10)
robot.program_abort()   #abort
time.sleep(3)
```

```
robot.logout()          #exit
```

4.5.9 Set rapid rate

Function	set_rapidrate(rapid_rate)
Description	Set robot rapid rate
Parameter	rapid_rate:Set robot rapid rate
Return value	Success: (0,) Failed: Others

4.5.10 Get rapid rate

Function	get_rapidrate()
Description	Get robot rapid rate
Parameter	
Return value	Success: (0, rapidrate) rapidrate is the speed rate, and Returned value is within a closed interval between 0 and 1 Failed: Others

4.6 Trajectory Reproduction

4.6.1 Set trajectory track configuration parameter

Function	set_traj_config(xyz_interval, rpy_interval, vel, acc)
Description	In setting the trajectory track configuration parameters, you can set the space position sample accuracy, pose sample accuracy, script execution running speed and script execution running acceleration.
Parameters	xyz_interval: Space position acquisition speed rpy_interval: Orientation capture accuracy vel: Script execution running speed acc: Script execution running acceleration
Return value	Success: (0,) Failed: Others

Sample Code：

```
import jkrc
import time

#coordinate system
COORD_BASE  = 0
COORD_JOINT = 1
COORD_TOOL  = 2
```

```
#motion mode
ABS = 0
INCR= 1

robot = jkrc.RC("192.168.2.160")
robot.login()
robot.power_on()
robot.enable_robot()
robot.joint_move(joint_pos =[1,1,1,1,1,1] ,move_mode = 0 ,is_block = True ,speed = 10 )
print("joint")
robot.set_traj_config([0.1, 0.1, 25, 100]) #Set track recurrence parameters, only the recording process is valid
time.sleep(0.1)
ret = robot.get_traj_config()#get trajectory recurrence parameters
print("traj_config:")
print(ret)
robot.set_traj_sample_mode(True, 'pythonTrack')#enable trajectory recurrence capture
time.sleep(0.1)
robot.joint_move(joint_pos =[-1,1,1,1,1,1] ,move_mode = 0 ,is_block = True ,speed = 30*3.14/180 )#blocking motion
robot.joint_move(joint_pos =[1,1,1,1,1,1] ,move_mode = 0 ,is_block = True ,speed = 30*3.14/180 )
# robot.jog(2,INCR,COORD_BASE,10,-2)
# robot.jog(2,INCR,COORD_BASE,10,2)
robot.set_traj_sample_mode(False, 'pythonTrack')# disable trajectory recurrence capture
time.sleep(1)
res = robot.generate_traj_exe_file('pythonTrack')# convert the captured trajectory files into scripts
print(res)
robot.program_load("track/pythonTrack")#loading trajectory programs
time.sleep(0.1)
robot.program_run()
```

4.6.2 Get trajectory track configuration parameters

Function	get_traj_config()
Description	By getting the trajectory track configuration parameters, you can get the space position sample accuracy, pose sample accuracy, script execution running speed, and script execution running acceleration.
Parameters	
Return value	Success: (0 , (xyz_interval, rpy_interval, vel, acc)) xyz_interval: Space position acquisition speed rpy_interval: Orientation capture accuracy vel: Script execution running speed acc: Script execution running acceleration Failed: Others

4.6.3 Set trajectory sample mode

Function	set_traj_sample_mode(mode, filename)
Description	Set trajectory sample mode.
Parameters	mode: Control mode, True means starting data collection, False means ending data

	collection. filename: The name of the file where data are stored.
Return value	Success: (0,) Failed: Others

4.6.4 Get trajectory sample state

Function	**get_traj_sample_status()**
Description	Get trajectory sample state. Note: It is not allowed to turn on the data collection switch again during the data collection process.
Parameters	
Return value	Success: (0,sample_status),sample_status: Data status, True means data are being collected, False means data collection has been finished. Failed: Others

4.6.5 Get exist trajectory file name

Function	**get_exist_traj_file_name ()**
Description	Get exist trajectory file name
Parameters	
Return value	Success: (0,) Failed: Others

4.6.6 Rename exist trajectory file name

Function	**rename_traj_file_name (src, dest)**
Description	Rename exist trajectory file name
Parameters	src: Source file name dest: The target file name, the length of the file name cannot exceed 100 characters, the file name cannot be empty, the target file name does not support Chinese
Return value	Success: (0,) Failed: Others

Sample Code：

```
import jkrc                    #import module
robot = jkrc.RC("192.168.2.226")    #returns a robot object
robot.login()
robot.rename_traj_file_name('\\home\\src', '\\home\\dst')
robot.logout() #exit
```

4.6.7 Remove the trajectory file in the controller

Function	**remove_traj_file(filename)**
Description	Remove the trajectory file in the controller
Parameters	filename: The name of the file to be deleted

Return value	Success: (0,) Failed: Others

Sample Code：

```
import jkrc                      #import module
robot = jkrc.RC("192.168.2.226")  #returns a robot object
robot.login()
robot.remove_traj_file('test')
robot.logout() #exit
```

4.6.8 Generate the trajectory execution script

Function	**generate_traj_exe_file(filename)**
Description	Generate the trajectory execution script
Parameters	filename: The filename of data
Return value	Success: (0,) Failed: Others

4.7 Robot kinematics

4.7.1 Convert Rpy to rotation matrix

Function	**rpy_to_rot_matrix(rpy = [rx,ry,rz])**
Description	Convert Rpy to rot matrix
Parameter	rpy: rpy parameters to be converted [rx,ry,rz]
Return value	Success: (0, rot_matrix). rot_matrix is a 3X3 rotation matirx Failed: Others

4.7.2 Convert rotation matrix to rpy

Function	**rot_matrix_to_rpy(rot_matrix)**
Description	Covert Rot matrix to rpy
Parameter	rot_matrix: Rot matrix data to be converted
Return value	Success: (0, rpy). rpy is a tuple, whose length is 3. The expression of rpy is (rx, ry, rz). Failed: Others

4.7.3 Convert quaternion to rotation matrix

Function	**quaternion_to_rot_matrix (quaternion = [w,x,y,z])**
Description	Convert Quaternion to rot matrix

Parameter	quaternion: Quaternion data to be converted
Return value	Success: (0, rot_matrix). rot_matrix is a 3*3 rotation matirx Failed: Others

4.7.4 Kine inverse

Function	**kine_inverse(ref_pos, cartesian_pose)**
Description	Calculate the kine inverse of the specified pose under the current tool, current installation angle, and current user coordinate frame settings
Parameter	ref_pos: Reference joint position for kine inverse cartesian_pose: Cartesian space pose value
Return value	Success: (0, joint_pos) joint_pos is a tuple with 6 elements (j1, j2, j3, j4, j5, j6), J1, J2, J3, J4, J5, and j6 represent the Angle values from joint 1 to joint 6 respectively Failure: Others

4.7.5 Kine forward

Function	**kine_forward(joint_pos)**
Description	Calculate the pose value of the specified joint position under the current tool, current installation angle and current user coordinate frame settings
Parameter	joint_pos: Joint space position
Return value	success: (0,Cartesian_pose), where Cartesian_pose is a tuple containing 6 elements (x,y,z,rx,ry,rz), with x,y,z,rx,ry,rz representing the pose value of robot tool end Failed: Others

Sample Code：

'''Kine forward'''

```
import jkrc
robot = jkrc.RC("192.168.2.64") #returns a robot object
robot.login()                #login
ret = robot.get_joint_position()
if ret[0] == 0:
    print("the joint position is :",ret[1])
else:
    print("some things happend,the errcode is: ",ret[0])
joint_pos = ret[1]
robot.kine_forward(joint_pos)   #Kine forward
robot.logout()  #exit
```

4.7.6 Convert rotation matrix to quaternion

Function	rot_matrix_to_quaternion(rot_matrix)
Description	Rot matrix to quaternion
Parameter	rot_matrix：3x3 Rot matrix to be converted
Return value	Success: (0, quaternion). quaternion is a tuple, whose length is 4. The expression of quaternion is (w, x, y, z). Failed: Others

Sample Code：

```
import jkrc
robot = jkrc.RC("192.168.2.160")
robot.login()
ret = robot.get_tcp_position()
print(ret)
rpy = [ret[1][3], ret[1][4], ret[1][5]]#get rpy
ret = robot.rpy_to_rot_matrix(rpy)#convert rpy to rot matrix
print(ret)
rot_matrix = ret[1]#get rot matrix
ret = robot.rot_matrix_to_rpy(rot_matrix)#convert rot matrix to rpy
print(ret)
ret = robot.rot_matrix_to_quaternion(rot_matrix)# convert rot matrix to quaternion
print(ret)
quaternion = ret[1]
ret = robot.quaternion_to_rot_matrix(quaternion)# convert rot matrix to quaternion
print(ret)
robot.logout()
```

4.8 Robot Servo Mode

4.8.1 Robot servo move enable

Function	servo_move_enable(enable)
Description	Robot servo move enable
Parameter	enable :TRUE means to enter the servo move control mode, FALSE means to quit the mode
Return value	Success: (0,) Failed: Others

4.8.2 Robot joint servo move extension

Function	servo_j(joint_pos, move_mode)

Description	Joint move control mode
Parameter	Please note the following items for the robot joint move control mode: (a) Before using this interface the user needs to call servo_move_enable(True) first to enter the position control mode (b) This command is generally used in trajectory planning in university research. (c) When users use this mode to control the robot motion, the controller planner will not be involved in the motion interpolation, and the position command will be sent to the servo directly. Therefore users need to do the trajectory planning by themselves. Otherwise the robot motion effectiveness can be poor, like violent shaking, which can not meet the user's expectation. (d) Since the control cycle time of the controller is 8ms, it is recommended that the user should send the command with a period of 8ms too, and continuously. There will be no effect if the command is sent only once. In case of a poor network, the command can be sent with a period less than 8ms. (e) The upper limit on the Jaka robot joint speed is 180 degrees per second. If the joint speed exceeds this limit due to the joint angle that is sent, this command will then fail. For example, if the joint angle that is sent is [1.5,0.5,0.5,0.5,0.5,0.5] (here the unit is degree) and the sending period is 8ms, thus 1.5/0.008 = 187.5 degrees per second, which exceeds the upper limit on the joint speed. Then the command will be invalid. (f) After using this command, the user needs to use servo_move_enable(False) to exit the position control mode. (g) There is a big difference between this command and the aforementioned joint_move(), which interpolation is processed by controller, and the user does not need to care about it. When using servo_j command, users need to make trajectory planning in advance. Otherwise the effect will be poor and can not meet the expectation. **If there is no special requirement, it is recommended to use joint_move instead of servo_j on robot joint move control.**
Parameters	joint_pos: target robot joint move position move_mode specified move mode: 0 for absolute move, 1 for incremental move
Return value	Success :(0,) Failed: Others

Sample Code：

'''using of servo_j'''

```
# -*- coding: utf-8 -*-
import sys
sys.path.append('D:\\vs2019ws\PythonCtt\PythonCtt')
import time
import jkrc
ABS = 0          # absolute motion
INCR = 1          # incremental motion
Enable = True
Disable = False
robot = jkrc.RC("192.168.2.160")#returns a robot object
robot.login()#login
robot.power_on() #power on
robot.enable_robot()
robot.servo_move_enable(Enable)  #enter position control mode
print("enable")
for i in range(200):
    robot.servo_j(joint_pos =[0.001,0,0,0,0,0.001],move_mode = INCR)#
for i in range(200):
    robot.servo_j(joint_pos =[-0.001,0,0,0,0,-0.001],move_mode = INCR)
robot.servo_move_enable(Disable)# exit position control mode
print("disable")
robot.logout()  #exit
```

4.8.3 Robot joint servo move extension

Function	servo_j_extend(joint_pos, move_mode, setp_num)
Description	(a) Please note the following items for the robot joint move control mode: (b) Before using this interface the user needs to call servo_move_enable(True) first to enter the position control mode (c) This command is generally used in trajectory planning in university research. (d) When users use this mode to control the robot motion, the controller planner will not be involved in the motion interpolation, and the position command will be sent to the servo directly. Therefore users need to do the trajectory planning by themselves. Otherwise the robot motion effectiveness can be poor, like violent shaking, which can not meet the user's expectation. (e) Since the control cycle time of the controller is 8ms, it is recommended that the user should send the command with a period of 8ms too, and continuously. There will be no effect if the command is sent only once. In case of a poor network, the command can be sent with a period less than 8ms. (f) The upper limit on the Jaka robot joint speed is 180 degrees per second. If the joint speed exceeds this limit due to the joint angle that is sent, this command will become invalid. For example, if the joint angle that is sent is [1.5,0.5,0.5,0.5,0.5,0.5] (here the unit is degree) and the sending period is 8ms, thus 1.5/0.008 = 187.5 degrees per second, which exceeds the upper limit on the joint speed. Then the command will be invalid. (g) After using this command, the user needs to use servo_move_enable(False) to exit the position control mode. (h) There is a big difference between this command and the aforementioned joint_move(), which interpolation is processed by controller, and the user does not need to care about it. When using servo_j command, users need to make trajectory

	planning in advance. Otherwise the effect will be poor and can not meet the expectation. If there is no special requirement, it is recommended to use joint_move instead of servo_j on robot joint move control.
Parameters	joint_pos: target robot joint move position move_mode specified move mode: 0 for incremental move, 1 for absolute move step_num: multiplying period, servo_j move period is step_num*8ms, where step_num>=1
Return value	Success: (0,) Failed: Others

4.8.4 Cartesian space servo mode

Function	**servo_p(cartesian_pose, move_mode)**
Description	Please note the following items for the robot Cartesian space servo mode: (a) Before using this interface the user needs to call servo_move_enable(True) first to enter the position control mode (b) This command is generally used in trajectory planning in university research. (c) When users use this mode to control the robot motion, the controller planner will not be involved in the motion interpolation, and after the controller kine inverse is calculated, the position command will be sent to the servo directly. Therefore users need to do the trajectory planning himself. Otherwise the robot motion effectiveness can be poor, like violent shaking, which cannot meet users' expectation. (d) Since the control cycle time of the controller is 8ms, it is recommended that the user should send the command with a period of 8ms too, and continuously. There will be no effect if the command is sent only once. In case of a poor network, the command can be sent with a period less than 8ms. (e) The upper limit on the Jaka robot joint speed is 180 degrees per second. If the joint speed exceeds this limit due to the position that is sent, this command will become invalid. (f) After using this command, the user needs to use servo_move_enable(False) to exit the position control mode. (g) There is a big difference between this command and the aforementioned linear_move(), which interpolation is processed by controller, and the user does not need to care about it. When using **servo_p** command, users need to make trajectory planning in advance. Otherwise the effect will be poor and can not meet the expectation. **If there is no special requirement, it is recommended to use linear_move instead of servo_p on robot Cartesian space move control.**
Parameters	Cartesian_pose: Target robot Cartesian space move position. move_mode: Specified move mode, 0 for absolute move, 1 for incremental move
Return value	Success: (0,) Failed: Others

Sample Code：
'''using of servo_p'''

```
# -*- coding: utf-8 -*-
import sys
sys.path.append('D:\\vs2019ws\PythonCtt\PythonCtt')
import time
import jkrc
PI=3.1415926

```

```
ABS = 0         # absolute motion
INCR = 1         # incremental motion
Enable = True
Disable = False

robot = jkrc.RC("192.168.2.160")#returns a robot object
robot.login()#login
robot.power_on() #power on
robot.enable_robot()
joint_pos=[PI/3,PI/3,-PI/3,PI/4,PI/4,0]
robot.joint_move(joint_pos,ABS,True,1)
robot.servo_move_enable(Enable)  # enter position control mode
print("enable")
for i in range(200):
    robot.servo_p(cartesian_pose = [1, 0, 0, 0, 0, 0],move_mode = INCR)
for i in range(200):
    robot.servo_p(cartesian_pose = [-1,0, 0, 0, 0, 0],move_mode = INCR)
robot.servo_move_enable(Disable)# exit position control mode
print("disable")
robot.logout()  #exit
```

4.8.5 Robot Cartesian space servo move extension

Function	**servo_p_extend(cartesian_pose, move_mode, step_num)**
Description	(a) Please note the following items for the robot Cartesian space servo mode: (b) Before using this interface the user needs to call servo_move_enable(True) first to enter the position control mode (c) This command is generally used in trajectory planning in university research. (d) When users use this mode to control the robot motion, the controller planner will not be involved in the motion interpolation, and after the controller kine inverse is calculated, the position command will be sent to the servo directly. Therefore users need to do the trajectory planning himself. Otherwise the robot motion effectiveness can be poor, like violent shaking, which cannot meet users' expectation. (e) Since the control cycle time of the controller is 8ms, it is recommended that the user should send the command with a period of 8ms too, and continuously. There will be no effect if the command is sent only once. In case of a poor network, the command can be sent with a period less than 8ms. (f) The upper limit on the Jaka robot joint speed is 180 degrees per second. If the joint speed exceeds this limit due to the position that is sent, this command will become invalid. (g) After using this command, the user needs to use servo_move_enable(False) to exit the position control mode. (h) There is a big difference between this command and the aforementioned linear_move(), which interpolation is processed by controller, and the user does not need to care about it. When using servo_p command, users need to make trajectory planning in advance. Otherwise the effect will be poor and can not meet the expectation. If there is no special requirement, it is recommended to use linear_move instead of servo_p on robot Cartesian space move control.
Parameters	Cartesian_pose: Target robot Cartesian space move position

	move_mode:Specify the movement mode,0 is absolute movement,1 is incremental movement step_num: multiplying period, servo_p move period is step_num*8ms, where step_num>=1
Return value	Success: (0,) Failed: Others

4.8.6 None filters in SERVO mode

Function	**servo_move_use_none_filter()**
Description	It is not allowed to use any filter in the robot SERVO mode. This command cannot be set in the SERVO mode. It can be set only after exiting the SERVO mode.
Parameters	
Return value	Success: (0,) Failed: Others

Sample Code：

```
import jkrc                    #import module
robot = jkrc.RC("192.168.2.226")  #returns a robot object
robot.login()
robot.servo_move_use_none_filter()
robot.logout() #exit
```

4.8.7 Use joint first-order low pass filter in SERVO mode

Function	**servo_move_use_joint_LPF(cutoffFreq)**
Description	Use Robot joint space first-order low-pass filter in SERVO mode. This command cannot be set in the SERVO mode. It can be set only after exiting the SERVO mode.
Parameters	cutoffFreq: First-order low-pass filter cut-off frequency.
Return value	Success: (0,) Failed: Others

Sample Code：

```
import jkrc                    #import module
robot = jkrc.RC("192.168.2.226")  #returns a robot object
robot.login()
robot.servo_move_use_joint_LPF(0.5)
robot.logout() #exit
```

4.8.8 Use joint nonlinear filter in SERVO mode

Function	**servo_move_use_joint_NLF(max_vr, max_ar, max_jr)**
Description	Use Robot joint space nonlinear filter in SERVO mode. This command can not be set in SERVO mode. It can be set only after exiting the SERVO mode.
Parameters	max_vr: The upper limit of Cartesian space orientation change speed (absolute value) °/s max_ar: The upper limit of accelerated speed of Cartesian space orientation change speed (absolute value)°/s^2 max_jr: The upper limit value of jerk (absolute value) of Cartesian space orientation change speed °/s^3

Return value	Success: (0,) Failed: Others

Sample Code：

```
import jkrc                    #import module
robot = jkrc.RC("192.168.2.226")  #returns a robot object
robot.login()
robot.servo_move_use_joint_NLF(max_vr=2, max_ar=2, max_jr=4)
robot.logout() #exit
```

4.8.9 Use Cartesian space nonlinear filter in SERVO mode

Function	**servo_move_use_carte_NLF(max_vp, max_ap, max_jp, max_vr, max_ar, max_jr)**
Description	Use Cartesian space nonlinear filter in the Robot Servo mode. This command can not be set in the SERVO mode. It can be set only after exiting the SERVO mode.
Parameters	max_vp: The upper limit (absolute value) of the move command speed in Cartesian space mm/s max_ap: The upper limit (absolute value) of the move command acceleration in Cartesian space mm/s^2 max_jp: The upper limit (absolute value) of the move command accelerated acceleration in Cartesian space mm/s^3 max_vr: The upper limit of Cartesian space orientation change speed (absolute value) °/s max_ar: The upper limit of accelerated speed of Cartesian space orientation change speed (absolute value)°/s^2 max_jr: The upper limit value of jerk (absolute value) of Cartesian space orientation change speed °/s^3
Return value	success: (0,) Failed: Others

Sample Code：

```
import jkrc                    #import module
robot = jkrc.RC("192.168.2.226")  #returns a robot object
robot.login()
robot.servo_move_use_carte_NLF(max_vp=2, max_ap=2, max_jp=4,max_vr=2, max_ar=2, max_jr=4)
robot.logout() #exit
```

4.8.10 Use joint multi-order mean filter in SERVO mode

Function	**servo_move_use_joint_MMF(max_buf, kp, kv, ka)**
Description	Use joint space multi-order mean filter in the Robot SERVO mode. This command can not be set in the SERVO mode. It can be set only after exiting the SERVO mode.
Parameters	max_buf: The buffer size of the mean filter. kp: Acceleration filter factor. kv: Speed filter factor ka: Position filter factor
Return value	Success: (0,) Failed: Others

Sample Code：

```
import jkrc                    #import module
robot = jkrc.RC("192.168.2.226")  #returns a robot object
robot.login()
robot.servo_move_use_joint_MMF(max_buf=20 , kp=0.2 , kv=0.4 ,ka=0.2)
robot.logout() #exit
```

4.8.11 Set the speed foresight parameters in SERVO mode

Function	**servo_speed_foresight (max_buf, kp)**
Description	Set the speed foresight parameters in SERVO mode
Parameters	max_buf: The buffer size of the mean filter. kp: Acceleration filter factor.
Return value	Success: (0,) Failed: Others

4.9 Force Control Robot Extensions

Additional force control sensors need to be loaded on the end of the tool for the following interfaces to work

4.9.1 Set sensor brand

Function	**set_torsensor_brand(sensor_brand)**
Description	Set the sensor brand. The number input represents the corresponding sensor brand, with the optional values of 1, 2 and 3, representing the different F/T sensors, respectively.
Parameters	sensor_brand: The number corresponding to the sensor brand
Return value	Success: (0,) Failed: Others

Sample Code：

```
import jkrc                    #import module
robot = jkrc.RC("192.168.2.165")  #returns a robot object
robot.login()
robot.set_torsenosr_brand(1)    #1 for SONY Semiconductor; 2 for Bosch Sensortec; 3 for ST Microelectronics
robot.logout() #exit
```

4.9.2 Get sensor brand

Function	**get_torsensor_brand()**
Description	Get the brand of the sensor currently being used. The numbers output represent the different sensor brands.
Parameters	
Return value	Success: (0,sensor_brand), where sensor_brand represents the sensor brand, with the optional values of 1, 2 and 3, representing the different F/T sensors, respectively. Failed: Others

Sample Code：

```
import jkrc                    #import module
robot = jkrc.RC("192.168.2.165")  #returns a robot object
```

```
robot.login()
ret=robot.get_torsenosr_brand()
if ret[0] == 1:
    print("the sensor_band is SONY Semiconductor")
elif ret[0] == 2:
     print("the sensor_band is BoschSensortec")
else:
    print("the sensor_band is ST Microelectronics")
robot.logout() #exit
```

4.9.3 Turn on or turn off F/T sensor

Function	set_torque_sensor_mode(sensor_mode)
Description	Turn on or turn off the F/T sensor currently being used.
Parameters	sensor_mode: The sensor working mode, 0 means turning off the F/T sensor, 1 means turning on the F/T sensor.
Return value	Success: (0,) Failed: Others

4.9.4 Set compliance control parameters

Function	set_admit_ctrl_config(axis, opt, ftUser, ftReboundFK, ftConstant, ftNormalTrack)
Description	When set the robot compliance control, you can set such parameters as joint axis numbers, compliance direction, damping force, rebound force, constant force and normal tracking.
Parameters	axis: represents the axis number of Cartesian space, with a value range of 0 to 5, representing the corresponding directions of fx, fy, fz, mx, my and mz, respectively. opt: The compliance direction, 0 for turning off, 1 for turning on. ftUser: The damping force, indicating that how much force the user needs to apply to make the robot move along a certain direction at the maximum speed. ftReboundFK: The rebound force, indicating the ability of the robot to return to its initial state. ftConstant: The constant force, all of which are set to 0 when manual operation. ftNormalTrack: The normal tracking, all of which are set to 0 when manual operation.
Return value	Success: (0,) Failed: Others

4.9.5 Start to identify the tool end payload

Function	start_torq_sensor_payload_identify(joint_pos)
Description	Start to identify the tool end payload, and input a tuple of 6 elements, representing the 6 joint angles of end position.
Parameters	joint_pos: the end position for the F/T sensor to do automatic payload identification, corresponding to 6 joint angles
Return value	Success: (0,) Failed: Others

Sample Code：

```
# -*- coding: utf-8 -*-
import sys
#sys.path.append('D:\\vs2019ws\PythonCtt\lib')
import time
import jkrc
PI = 3.1415926

robot = jkrc.RC("10.5.5.100")#returns a robot object
ret = robot.login()#login
ret = robot.power_on()
ret = robot.enable_robot()
robot.set_torsenosr_brand(2)
robot.set_torque_sensor_mode(1)
robot.set_compliant_type(1, 1)
print("inint sensor comple")
print("ready to run")
ret = robot.get_joint_position()
joint_pos_origin = ret[1]
joint_pos = ret[1]
print(joint_pos)
joint_pos[3] += PI / 4
if (joint_pos[3] > 265 * 180 / 180):
    joint_pos[3] -= PI/2
joint_pos[4] += PI / 4
if (joint_pos[4] > 320 * 180 / 180):
    joint_pos[4] -= PI/2
joint_pos[5] += PI / 4
if (joint_pos[5] > 265 * 180 / 180):
    joint_pos[5] -= PI
print(joint_pos)
ret = robot.start_torq_sensor_payload_identify(joint_pos)
time.sleep(1)
flag = 1
while (1 == flag):
    ret = robot.get_torq_sensor_identify_staus()
    print(ret)
    time.sleep(1)
    flag = ret[1]
print("identy_finish")
ret = robot.get_torq_sensor_payload_identify_result()
print(ret)
ret = robot.set_torq_sensor_payload()
print(ret)
ret = robot.get_torq_sensor_payload_identify_result()
print(ret)
robot.joint_move(joint_pos_origin,0,1,10)
print("back")
robot.logout()  #exit
```

4.9.6 Get end payload identification status

Function	get_torq_sensor_identify_status()
Description	Get end payload identification status
Parameters	identify_status: The identification status, 0 means identification completed, 1 means identification uncompleted, 2 means identification failed.
Return value	Success: (0,**identify_status**) Failed: Others

4.9.7 Get end payload identification result

Function	get_torq_sensor_payload_identify_result()
Description	Get the end load identification result, and input load quality and centroid coordinates
Parameters	mass: payload mass, unit: kg centroid: load centroid coordinates [x, y, z], unit: mm
Return value	Success: (0,) Failed: Others

4.9.8 Set sensor end load

Function	set_torq_sensor_tool_payload (mass, centroid)
Description	Set the sensor end load, and input load quality and centroid coordinates.
Parameters	mass: payload mass, unit: kg centroid: load centroid coordinates [x, y, z], unit: mm
Return value	Success: (0,) Failed: Others

Sample Code：

```
import jkrc                        #import module
robot = jkrc.RC("192.168.2.226")     #returns a robot object
robot.login()
robot.set_torq_sensor_tool_payload(mass= 1, centroid =[10,10,0])
robot.logout() #exit
```

4.9.9 Get sensor end load

Function	get_torq_sensor_tool_payload ()
Description	Get sensor end load quality and centroid coordinates.
Parameters	
Return value	Success: (0,(mass,centroid)), mass: The load quality, unit: kg centroid: load centroid coordinates [x, y, z], unit: mm Failed: Others

4.9.10 Force-control admittance enable control

Function	enable_admittance_ctrl (enable_flag)
Description	Force-control admittance enable control
Parameters	enable_flag: The flag, 0 means turning off force-control drag enable, 1 means turning on force-control admittance enable.
Return value	Success: (0,) Failed: Others

Sample Code：

```python
# -*- coding: utf-8 -*-
import sys
#sys.path.append('D:\\vs2019ws\PythonCtt\lib')
import time
import jkrc
PI=3.1415926

robot = jkrc.RC("10.5.5.100")#returns a robot object
ret = robot.login()#login
ret = robot.power_on()
ret = robot.enable_robot()
robot.set_torsenosr_brand(2)
robot.set_torque_sensor_mode(1)
robot.set_compliant_type(1, 1)
print("inint sensor comple")
print("ready to run")
#Set compliance control parameters
ret = robot.set_admit_ctrl_config(0, 0, 20, 5, 0, 0)
ret = robot.set_admit_ctrl_config(1, 0, 20, 5, 0, 0)
ret = robot.set_admit_ctrl_config(2, 1, 20, 10, 0, 0)
ret = robot.set_admit_ctrl_config(3, 0, 20, 5, 0, 0)
ret = robot.set_admit_ctrl_config(4, 0, 20, 5, 0, 0)
ret = robot.set_admit_ctrl_config(5, 0, 20, 5, 0, 0)
#Set force control drag enable, I means on and 0 means off
ret = robot.enable_admittance_ctrl(1)
print("enable_admittance_ctrl open！ ")
print("input any word to quit:")
a = input()
ret = robot.enable_admittance_ctrl(0)
ret = robot.set_admit_ctrl_config(2, 0, 20, 5, 0, 0)
robot.set_torque_sensor_mode(0)
robot.logout()  #exit
```

4.9.11 Set the force control type and sensor initial state

Function	set_compliant_type(sensor_compensation, compliance_type)
Description	Set the force control type and sensor initial state
Parameters	sensor_compensation: Turn on sensor compensation enable flag. 1 means turning on initialization, and 0 means no initialization

	compliance_type: The force control type. 0 means not using any kind of compliance control method, 1 means using constant compliance control, and 2 means using speed compliance control
Return value	Success: (0,) Failed: Others

Sample Code：

```
import jkrc                          #import module
robot = jkrc.RC("192.168.2.226")     #returns a robot object
robot.login()
robot.set_compliant_type(1,1)
robot.logout() #exit
```

4.9.12 Get the force control type and sensor initial state

Function	**get_compliant_type()**
Description	Get the force control type and sensor initial state
Parameters	
Return value	Success: (0, sensor_compensation, compliance_type) sensor_compensation: Turn on sensor compensation enable flag. 1 means turning on initialization, and 0 means no initialization compliance_type: The force control type. 0 means not using any kind of compliance control method, 1 means using constant compliance control, and 2 means using speed compliance control Failed: Others

4.9.13 Get the force-control compliance control parameters

Function	**get_admit_ctrl_config()**
Description	By getting the force-control compliance control parameters, you can get the compliance direction, damping force, rebound force, constant force and normal tracking which are corresponding to the 6 joints.
Parameters	
Return value	Success: (0, [[opt, ftUser, ftReboundFK, ftConstant, ftNormanlTrack], ……]) opt: The compliance direction, with an optional value range of 1 to 6, corresponding to the directions of fx, fy fz, mx, my and mz, respectively, and 0 means no check. ftUser: The damping force, indicating that how much force the user needs to apply to make the robot move along a certain direction at the maximum speed. ftReboundFK: The rebound force, indicating the ability of the robot to return to its initial state. ftConstant: The constant force, all of which are set to 0 when manual operation. ftNormalTrack: The normal tracking, all of which are set to 0 when manual operation. Failed: Others

4.9.14 Set the F/T sensor IP address

Function	set_torque_sensor_comm(type, ip_addr, port)
Description	Set the F/T sensor IP address.
Parameters	type: The sensor type ip_addr: The sensor IP address port: The port number
Return value	Success: (0,) Failed: Others

4.9.15 Get the F/T sensor IP address

Function	get_torque_sensor_comm()
Description	Get the F/T sensor IP address.
Parameters	
Return value	Success: (0,ip_addr),ip_addr: The sensor IP address. Failed: Others

4.9.16 Disable the torque control

Function	disable_force_control ()
Description	Disable the torque control
Parameters	
Return value	Success: (0,) Failed: Others

4.9.17 Set speed compliance control parameters

Function	set_vel_compliant_ctrl (level, rate1, rate2, rate3, rate4)
Description	Set the speed compliance control parameters. There are 3 speed compliance control levels and 4 rate levels.
Parameters	level: The compliance control levels which are divided into level 1, 2, and 3 1, 2, 3 rate1: Rate level 1 rate2: Rate level 2 rate3: Rate level 3 rate4: Rate level 4 Note: Relations between rate levels: 0<rate4<rate3<rate2<rate1<1; (a) When level = 1, only the values of rate1 and rate 2 can be set, and the values of rate 3 and rates 4 are all zero (b) When level =2, only the values of rate1, rate 2 and rate 3 can be set, and the value of rate 4 is zero (c) When level = 3, all the values of rate 1, rate 2, rate 3 and rate 4 can be set

Return value	Success: (0,) Failed: Others

4.9.18 Set the compliance control torque conditions

Function	**set_compliance_condition (fx, fy, fz, tx, ty, tz)**
Description	Set the compliance control torque conditions.
Parameters	fx: the force along X-axis, unit: N fy: the force along Y-axis, unit: N fz: the force along Z-axis, unit: N tx: the torque around X-axis, unit: Nm ty: the torque around Y-axis, unit: Nm tz: the torque around Z-axis, unit: Nm
Return value	Success: (0,) Failed: Others

4.9.19 Set the low-pass filter parameters of the force control

Function	**set_torque_sensor_filter(HZ);**
Description	Set condition of compliance control torque
Parameters	HZ: low-pass filter parameters，unit：HZ
Return value	Success: (0,) Failed: Others

4.9.20 Get the low-pass filter parameters of the force control

Function	**get_torque_sensor_filter();**
Description	Set condition of compliance control torque
Parameters	HZ: low-pass filter parameters，unit：HZ
Return value	Success: (0, HZ) Failed: Others

4.9.21 Set the sensor limit parameter configuration of the sensor

Function	**set_torque_sensor_soft_limit(fx, fy, fz, tx, ty, tz);**
Description	Set condition of compliance control torque
Parameters	fx: force along x-axis, unit: N fy: force along the y-axis, unit: N fz: force along the z-axis, unit: N tx: torque around the x-axis, unit: Nm ty: torque around y-axis, unit: Nm tz: torque around z-axis, unit: Nm
Return value	Success: (0,) Failed: Others

4.9.22 Get the sensor limit parameter configuration of the sensor

Function	**get_torque_sensor_soft_limit();**
Description	Set condition of compliance control torque
Parameters	
Return value	Success: (0,(fx,fy,fz,tx,ty,tz)) fx: force along x-axis, unit: N fy: force along the y-axis, unit: N fz: force along the z-axis, unit: N tx: torque around the x-axis, unit: Nm ty: torque around y-axis, unit: Nm tz: torque around z-axis, unit: Nm Failed: Others

4.9.23 Zero calibration of sensor

Function	**Zero_end_sensor();**
Description	Zero calibration of end sensor
Parameters	None
Return value	Success: (0,) Failed: Others

4.9.24 Get tool drive state

Function	**Get_tool_drive_state();**
Description	Get the tool drive coordinate system
Parameters	None
Return value	Success: (0,enable_flag,drive_state) Enable_flage: 0 is to turn off force control dragging, 1 is to turn it on, drive_stat is whether the current state of dragging triggers singularity point, speed, joint limit warning Failed: Others

4.9.25 Get the force sensor's coordinate system in drag mode

Function	get_tool_drive_frame();
Description	Get the tool drive coordinate system
Parameters	None
Return value	Success: (0,ftFrame) ftFrame: 0 Tool 1 World Failed: Others

4.9.26 Set the force sensor's coordinate system in drag mode

Function	set_tool_drive_frame(ftFrame);
Description	set the tool drive coordinate system
Parameters	None
Return value	Success: (0,ftFrame)

	ftFrame: 0 Tool 1 World Failed: Others

4.9.27 Get fusion drive sensitivity

Function	get_fusion_drive_sensitivity_level();
Description	Get the fusion drive sensitivity level
Parameters	None
Return value	Success: (0,level) Level: sensitivity level, value:0-5, 0 means not turned on Failed: Others

4.9.28 Set fusion drive sensitivity

Function	Set_fusion_drive_sensitivity_level(level);
Description	Set the fusion drive sensitivity level
Parameters	Level: sensitivity level, value:0-5, 0 means not turned on
Return value	Success: (0,) Failed: Others

4.9.29 Get motion limit (singularity and joint limit) warning range

Function	get_motion_limit_warning_range(warningRange);
Description	Get the warning range of motion limit (singularity and joint limit)
Parameters	None
Return value	Success: (0,warningRange) WarningRange: the range that warnings would happen Failed: Others

4.9.30 Set motion limit (singularity and joint limit) warning range

Function	set_motion_limit_warning_range(warningRange);
Description	set the warning range of motion limit (singularity and joint limit)
Parameters	WarningRange: the range that warnings would happen
Return value	Success: (0,) Failed: Others

4.9.31 Get compliant speed limit

Function	**get_compliant_speed_limit();**
Description	Get the compliant speed limit
Parameters	None
Return value	Success: (0,vel,angularvel) Vel: linear velocity limit, mm/s Angularvel: angular velocity limit, rad/s Failed: Others

4.9.32 Set compliant speed limit

Function	set_compliant_speed_limit(vel，angularvel);
Description	set the compliant speed limit
Parameters	Vel: linear velocity limit, mm/s Angularvel: angular velocity limit, rad/s
Return value	Success: (0,) Failed: Others

4.9.33 Get torque reference center

Function	get_torque_ref_point();
Description	Get the torque reference center
Parameters	None
Return value	Success: (0,refpoint) Refpoint: 0 torque sensor center, 1 TCP center Failed: Others

4.9.34 Set torque reference center

Function	set_torque_ref_point(refpoint);
Description	set the torque reference center
Parameters	Refpoint: 0 torque sensor center, 1 TCP center
Return value	Success: (0,) Failed: Others

4.9.35 Get end sensor sensitivity

Function	get_end_sensor_sensitivity_threshold ();
Description	Get the end sensor sensitivity threshold
Parameters	
Return value	Success: (0,data) Fx: force on x axis, unit: N Fy: force on y axis, unit: N Fz: force on z axis, unit: N Tx: torque around x axis, unit: Nm Ty: torque around y axis, unit: Nm Tz: torque around z axis, unit: Nm Failed: Others

4.9.36 Set end sensor sensitivity

Function	set_end_sensor_sensitivity_threshold (fx, fy, fz, tx, ty, tz);
Description	set the end sensor sensitivity threshold
Parameters	Fx: force on x axis, unit: N Fy: force on y axis, unit: N Fz: force on z axis, unit: N Tx: torque around x axis, unit: Nm Ty: torque around y axis, unit: Nm Tz: torque around z axis, unit: Nm An array of six element, the larger the value is, the less sensitive the sensor is
Return value	Success: (0,data) Failed: Others

4.10 FTP Services

4.10.1 Initialize FTP

Function	init_ftp_client()
Description	Initialize the FTP client, establish connection with control cabinet, and import and export program and track
Parameters	none
Return value	Success: (0,) Failed: Others

4.10.2 Close FTP client

Function	close_ftp_client()
Description	Close FTP client
Parameters	none
Return value	success: (0,) Failed: Others

4.10.3 Interrogate the directory of controller FTP

Function	get_ftp_dir(remotedir, type);
Description	Interrogate controller directory
Parameters	remotedir: The controller internal folder name type: 0 for file and folder, 1 for file, 2 for folder
Return value	Success: (0,ret), where ret is a string Failed: Others

Sample Code：

```
import jkrc
robot = jkrc.RC("192.168.2.26")
robot.login()
dir= "/program/"
#login controller，you need to convert 192.168.2.26 into your own controller IP
robot.init_ftp_client()
result = robot.get_ftp_dir("/program/", 0)
print(result)
robot.close_ftp_client()
robot.logout()
```

4.10.4 Download FTP file

Function	download_file(local, remote, opt)
Description	Download the file or folder from Robot control cabinet FTP to your local directory, and Interrogate track "/track/" and script program "/program/"
Parameters	remote: the controller internal filename absolute path. Whether the folder name should

	be ended with a "\" or "/" should depend on the system. For example, a path of "/program/test/test.jks" for a single file, or a path of "/program/test/" for a folder local: the absolute filename path to download the file to local directory opt: 1 for single file, 2 for folder
Return value	Success: (0,) Failed: Others

Sample Code: Download the program folder on the ftp to the program folder on the desktop

```
import jkrc
robot = jkrc.RC("192.168.2.26")#VMmodel
robot.login()
remote = "/program/"
local= "C:\\Users\\Administrator\\Desktop\\program\\track\\"
robot.init_ftp_client()
result = robot.download_file(local, remote, 2)
print(result)
robot.close_ftp_client()
robot.logout()
```

4.10.5 Upload files to FTP

Function	**upload_file(local, remote, opt)**
Description	Upload files with specified types and names from your local directory to controller
Parameters	remote: the absolute filename path to upload the file to controller. Whether a folder name should be ended with a "\" or "/" should depend on the system local: the local filename absolute path opt: 1 for single file, 2 for folder
Return value	Success: (0,) Failed: Others

Sample Code: Upload all files and folders from the lxxpro folder on the desktop to the program/folder of the ftp

```
import jkrc
robot = jkrc.RC("192.168.2.26")#VMmodel
robot.login()
remote = "/program/"
local= "C:\\Users\\Administrator\\Desktop\\lxxpro\\"
robot.init_ftp_client()
result = robot.upload_file(local, remote, 2)
print(result)
robot.close_ftp_client()
robot.logout()
```

4.10.6 Rename FTP files

Function	**upload_file(local, remote, opt)**
Description	Rename controller files with specified types and names
Parameters	remote: the controller internal filename absolute path. Whether the folder name should be ended with a "\" or "/" should depend on the system.

	des: name of target files renamed opt: 1 for single file, 2 for folder. When renaming files, all files in the folder will be renamed to be used by /track/ easily
Return value	Success: (0,) Failed: Others

Sample Code: Rename all files and folders in the ftp's lxxpro folder to lxx

```
import jkrc

robot = jkrc.RC("192.168.2.26")#VMmodel
robot.login()
remote = "/lxxpro/"
des = "lxx"
robot.init_ftp_client()
result = robot.rename_ftp_file(remote, des, 2)
print(result)
robot.close_ftp_client()
robot.logout()
```

4.10.7 Delete FTP files

Function	**del_ftp_file(remote, opt)**
Description	Upload files with specified types and names from controller to your local directory
Parameters	remote: the controller internal filename absolute path. Whether the folder name should be ended with a "\" or "/" should depend on the system. opt: 1 for single file, 2 for folder
Return value	Success: (0,) Failed: Others

Sample Code: Be careful, the demo will delete all programs！！！！

```
import jkrc

robot = jkrc.RC("192.168.2.26")#VMmodel
robot.login()
dir= "/program/"
robot.init_ftp_client()
result = robot.del_ftp_file("/program/", 2)
print(result)
robot.close_ftp_client()
robot.logout()
```

5.Feedback and Error Correct

If there is an inaccurate description or error in the document, the reader is kindly requested to point the finger at criticism. If you find any questions or comments you would like to make during your reading, you can send an email to support@jaka.com and our colleagues will reply.