JAKA Robotics

TCP Protocol

Original Instructions (en)

File Version: 2.0.4

Notice:

Manual Instruction

The main content of this manual is the interface description and interface command format under the JAKA TCP protocol.

This manual is aimed at users who should have received basic programming training, which will be more helpful in the use of robots.

More Information

For more product information, scan the QR code on the right to visit our official website: www.jakarobotics.com.

Contents

Chapter 1 User Development Instruction

Develop robot custom software according to JAKA TCP protocol;

* Start a TCP server to receive user-defined commands;

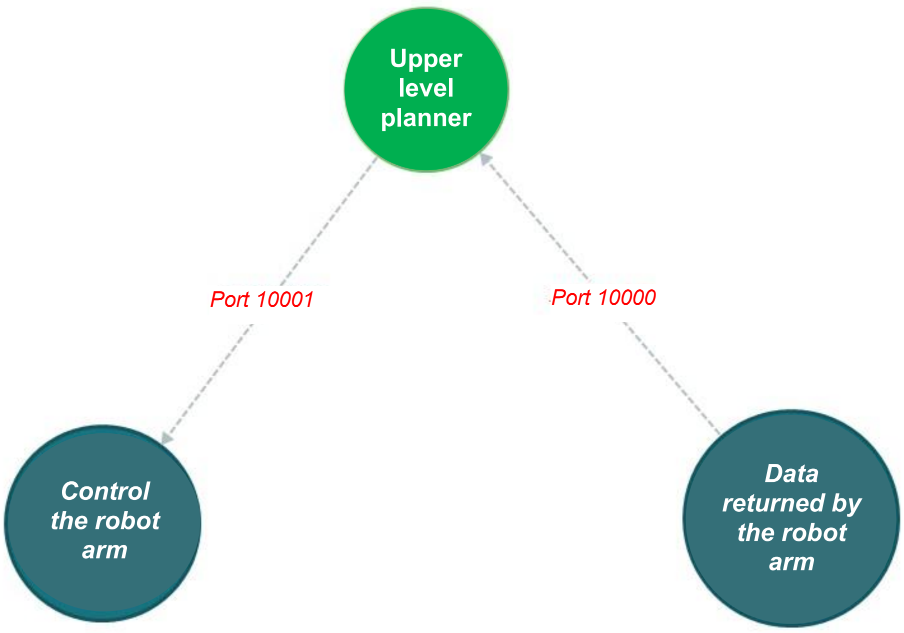
* The user needs to implement the corresponding TCP client call according to this protocol, that is, the user needs to design the upper level planner to send the robot data;

* This function can be used simultaneously with JAKA-APP without disconnecting the APP from the controller;

* Server listening port number: 10001， used to send control commands to the robot;

* Server listening port number: 10000, used to receive the data returned by the robot;

* **As it is difficult to use this protocol for custom software development, the JAKA SDK is recommended for custom software development if there is no special requirements.**



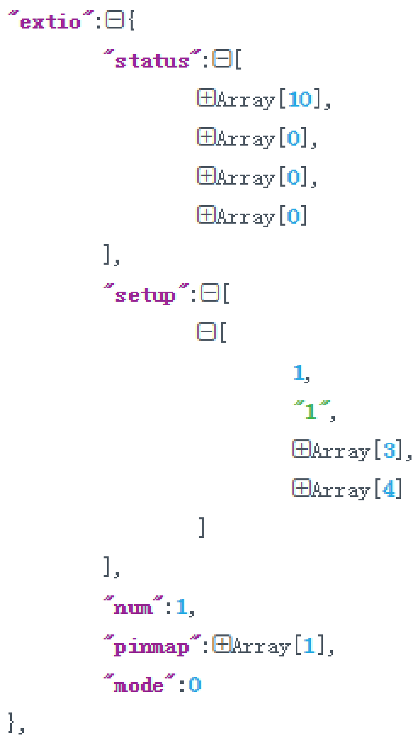
Chapter 2 Port 10000 Description

The overview of the data structure is shown as below.

```
{
   "len":3547,
   "drag_status":false,
   "extio":Object{...},
   "errcode":"0x0",
   "errmsg":"",
   "monitor_data":Array[6],
   "torqsensor":Array[2],
   "joint_actual_position":Array[6],
   "actual_position":Array[6],
   "din":Array[144],
   "dout":Array[144],
   "ain":Array[66],
   "aout":Array[66],
   "tio_din":Array[8],
   "tio_dout":Array[8],
   "tio_ain":Array[1],
   "task_state":4,
   "homed":Array[9],
   "task_mode":1,
   "interp_state":0,
   "enabled":true,
   "paused":false,
   "rapidrate":1,
   "current_tool_id":0,
   "current_user_id":0,
   "on_soft_limit":1,
   "emergency_stop":0,
   "drag_near_limit":Array[6],
   "funcdi":Array[15],
   "powered_on":1,
   "inpos":true,
   "motion_mode":1,
   "curr_tcp_trans_vel":0,
   "protective_stop":0,
   "point_key":0,
   "netState":1
}
```

Description:

1. **len**: the length of the data packet, generally the maximum length does not exceed 5000 bytes.
2. **drag_status**: Boolean value to indicate whether it is in the drag mode.
3. **extio**: the configuration and status of the extended IO modules, with internal data structure as below.



```
"extio":⊟{
        "status":⊟[
                ⊞Array[10],
                ⊞Array[0],
                ⊞Array[0],
                ⊞Array[0]
        ],
        "setup":⊟[
                ⊟[
                        1,
                        "1",
                        ⊞Array[3],
                        ⊞Array[4]
                ]
        ],
        "num":1,
        "pinmap":⊞Array[1],
        "mode":0
},
```

(1) **status:** status of all extended IO pins, including DI, DO, AI, AO.

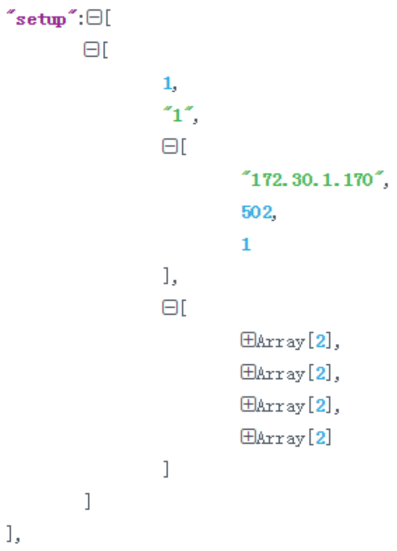
status[0] the status of all extended DIs,

status[1] the status of all extended DOs,

status[2] the status of all extended AIs,

status[3] the status of all extended AOs.

(2) **setup**: array of all extended IO configurations, 8 maximum. For each extended IO module, the setup follows the structure below.



```
"setup":⊟[
        ⊟[
                1,
                "1",
                ⊟[
                        "172.30.1.170",
                        502,
                        1
                ],
                ⊟[
                        ⊞Array[2],
                        ⊞Array[2],
                        ⊞Array[2],
                        ⊞Array[2]
                ]
        ]
],
```

setup[N][0]: communication type, 0:Modbus RTU, 1:ModbusTCP/IP.

setup[N][1]: the alias name of the extended IO module.

setup[N][2]: communication configuration, which varies as per communication type.

① For Modbus RTU (setup[N][0] == 0):

setup[N][2][0]: bard rate

setup[N][2][1]: data bits length

setup[N][2][2]: slave ID

setup[N][2][3]: data bits

setup[N][2][4]: parity check bit

setup[N][2][5]: stop bits length

② For Modbus TCP (setup[N][0] == 1) :

setup[N][2][0]: IP address

setup[N][2][1]: port

setup[N][2][2]: slave ID.

(3) **setup[N][3]:**pin configuration

setup[N][3][0][0]: starting offset of the register in DI

setup[N][3][0][1]: number of DI

setup[N][3][1][0]: starting offset of the register in DO

setup[N][3][1][1]: number of DO

setup[N][3][2][0]: starting offset of the register in AI

setup[N][3][2][1]: number of AI

setup[N][3][3][0]: starting offset of the register in AO

setup[N][3][3][1]: number of AO

(4) **num:** the number of extended IO module

(5) **pinmap:** array of the pin map of the extended IO module

pinmap[N][0]: starting offset of the register in DI for setup[N]

pinmap[N][1]: starting offset of the register in DO of setup[N]

pinmap[N][2]: starting offset of the register in AI of setup[N]

pinmap[N][2]: starting offset of the register in AO of setup[N]

(6) **mode**: status of the extended IO module, 0 for status on and 1 for off.

4. **errcode**: error code of from controller in hex string, like "0x0".
5. **errmsg**: error message string from controller corresponding to the errcode.
6. **monitor_data**: monitoring data of robot and cabinet for diagnosis.

monitor_data[0]: SCB major version number

monitor_data[1]: SCB minor version number

monitor_data[2]: Control cabinet temperature

monitor_data[3]: Control cabinet bus average power

monitor_data[4]: Control cabinet bus average current

monitor_data[5]: Monitoring data of joint axes 1-6 of the robot in array

monitor_data[5][0]: instantaneous current of the robot axis, monitor_data[5][1]: instantaneous voltage of the robot axis,

monitor_data[5][2]: instantaneous temperature of the robot axis，

monitor_data[5][3]: instant average power,

monitor_data[5][4]: current fluctuation,

monitor_data[5][5]: cumulative running circles,

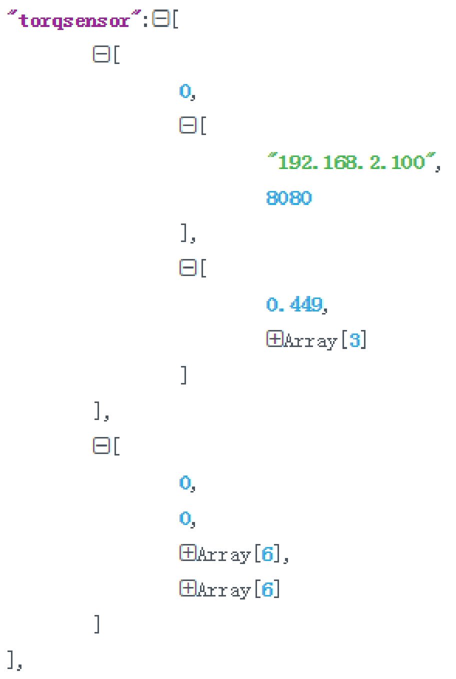
monitor_data[5][6]: cumulative running time,

monitor_data[5][7]: running circles after this boot,

monitor_data[5][8]: running time after this boot,

monitor_data[5][9]: joint torque

7. **torqsensor**: array of the torque setup and status, with the data structure below.



```
"torqsensor":⊟[
        ⊟[
                0,
                ⊟[
                        "192.168.2.100",
                        8080
                ],
                ⊟[
                        0.449,
                        ⊞Array[3]
                ]
        ],
        ⊟[
                0,
                0,
                ⊞Array[6],
                ⊞Array[6]
        ]
],
```

(1) torqsensor[0]: setup of current torque sensor

① torqsensor[0][0]: communication type, 0 for TCP/IP communication, 1 for serial port communication;

② torqsensor[0][1]: communication configuration of the torque sensor. For communication with TCP/IP address, it will be [IP, port], for communication with serial port, it will be [baudrate, databits, stopbits, parity].

③ torqsensor[0][2]: mass of the payload at the end of the torque sensor, and center position of the payload.

(2) torqsensor[1]: status of the torque sensor

① torqsensor[1][0]: torque sensor status, 1 for working, 0 for off.

② torqsensor[1][1]: torque sensor error code.

③ torqsensor[1][2]: torque sensor actual contact force (6 dimensions).

④ torqsensor[1][3]: torque sensor original reading value (6 dimensions).

8. **joint_actual_position**: actual position of all 6 joints.

9. **actual_position**: actual position of the robot TCP in Cartesian space.

10. **din**: array of the status of all the digital inputs in cabinet.

11. **dout**: array of the status of all the digital outputs in cabinet.

12. **ain**: array of the status of all the analog inputs in cabinet.
13. **aout**: array of the value of all the analog outputs in cabinet.
14. **tio_din**: array of the status of all the tool digital inputs.
15. **tio_dout**: array of the status of all the tool digital outputs.
16. **tio_ain**: array of the status of all the tool analog inputs.
17. **task_state**: indicator of the power status and enabling status, 1: the robot is powered off, 2: the robot is powered on, 3: the robot is enabled, 4: the robot is enabled.
18. **homed**: home status of each joint (obsolete).
19. **task_mode**: task mode of the robot. 1: manual mode, 2: automatic mode, 3: reserved, 4: drag mode.
20. **interp_state**: program status, 0: idle, 1: loading, 2: paused, 3: running.
21. **enabled**: Boolean value to show robot enabling status.
22. **paused**: Boolean value to show program pausing status.
23. **rapidrate**: the scale rate of the robot movement.
24. **current_tool_id**: index of current TCP.
25. **current_user_id**: index of current user coordinate frame.
26. **on_soft_limit**: Boolean status to show whether it is on the soft limit now.
27. **emergency_stop**: status of the emergency stop in system, 0 for estop reset, 1 for estop triggered.
28. **drag_near_limit**: status to show whether it is dragged to the limit position, 0 is no, 1 is yes.
29. **funcdi**: setup array of the function DIs. For each function DI, the setup will be in format [diType, diIndex]. For diType, 0 indicates the cabinet DI, 1 indicates the Tool DI and 2 indicates the extended DI.

    funcdi[0]: function DI setup to start the program.

    funcdi[1]: function DI setup to pause the program.

    funcdi[2]: function DI setup to continue the program.

    funcdi[3]: function DI setup to stop the program.

    funcdi[4]: function DI setup to power on the robot.

    funcdi[5]: function DI setup to power off the robot.

    funcdi[6]: function DI setup to enable robot.

    funcdi[7]: function DI setup to disable robot.

    funcdi[8]: function DI setup to level-1 reduction mode.

    funcdi[9]: function DI setup to trigger protective stop.

    funcdi[10]: function DI setup to return to safety position.

    funcdi[11]: function DI setup to level-2reduction mode.

    funcdi[12]: function DI setup to clear error code.

    funcdi[13]: function DI setup to enter drag mode.

    funcdi[14]: function DI setup to exit drag mode.
30. **powered_on**: power status of the robot, 0 for powered-off, 1 for powered-on.
31. **inpos**: indicator to show whether it's in position. True for robot is still, false for in movement.
32. **motion_mode**: motion mode (obsolete).
33. **curr_tcp_trans_vel**: the current movement speed at the end of current TCP.
34. **protective_stop**: protective stop status, 0 for normal and 1 protective stopped.

35. **point_key**: status to indicate whether the point button at the end of the robot is pushed down, 0 for button pushed down, and non-zero for otherwise.
36. **netState**: socket connection state (obsolete).

Chapter 3 Instruction Interface Format

Message format: All messages are in JSON string format;

* Format of message sent:

{"cmdName":"cmd","parameter1":"parameter1","parameter2":"parameter2",...}

* Format of message received:

{"cmdName":"cmd","errorCode":"0","errorMsg":"unknown","parameter1":"parameter1","parameter2":"parameter2"}

Chapter 4 Available interface(s)

4.1 Power on

Send message: {"cmdName": "power_on"}

Receive message: {"errorCode": "0", "errorMsg": "", "power status": "True", "cmdName": "power_on"}

4.2 Power off

Send message: {"cmdName": "power_off"}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "power_off"}

4.3 Enable robot

Send message: {"cmdName":"enable_robot"}

Receive message: {"errorCode": "0", "enabled status": "True", "cmdName": "enable_robot", "errorMsg": ""}

4.4 Disable the robot

Send message: {"cmdName":"disable_robot"}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "disable_robot"}

4.5 Shut down the robot and controller

Send message: {"cmdName":"shutdown"}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "shutdown"}

4.6 Jog command

Jog commands include the single-axis movement in joint space and the **single-axis movement** in cartesian space

jog_mode stands for move mode, and there are 4 optional values:

Jog_stop (Jog stops):	0
Continue (continuous motion):	1
Increment (step motion):	2
ABS (absolute motion):	3

coord_map stands for the options of coordinate frame, and there are 3 options:

Move in world coordinate frame:	0
Move in joint space:	1
Move in tool coordinate frame:	2

jnum stands for the axis number in joint space, and the axis numbers from axis 1 to axis 6 correspond to 0 to 5 respectively

jnum stands for x, y, z, rx, ry, rz in cartesian space, corresponding to number 0 to 5 respectively

jogvel stands for velocity

poscmd stands for the stepping value, unit of single joint move is deg, and unit of space single axial move is mm

When the value of **jog_mode is 0**, there are only 3 parameters

Send message: {"cmdName":"jog","jog_mode":0, "coord_map":1, "jnum":1}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "jog"}

Description:

(a) The command sent means the robot stops the jog movement of robot joint 2 in joint space

When the value of **jog_mode is 1**, there are only 4 parameters

Send message: {"cmdName":"jog","jog_mode":1, "coord_map":1, "jnum":1, "jogvel":30}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "jog"}

Description:

(a) The command sent means the robot joint 2 is doing jog movement at 30 deg/s in joint space

(a) Note that the positive and negative direction of the movement is reflected by the positive and negative value of jogvel.

(b) Before sending the motion command, stop the motion of current joint.

When the value of **jog_mode is 2**, there are only 5 parameters

Send message: {"cmdName":"jog","jog_mode":2, "coord_map":1, "jnum":3, "jogvel":30, "poscmd":20}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "jog"}

Description:

(a) The command sent means the robot joint 2 moves 20 deg in the positive direction at 30 deg/s in joint space.

(b) Note that the positive and negative direction of the movement can be reflected by either the positive and negative value of jogvel or the positive and negative value of poscmd, which is recommended.

When the value of **jog_mode is 3**, there are only 5 parameters.

Send message: {"cmdName":"jog","jog_mode":3, "coord_map":1, "jnum":3, "jogvel":30, "poscmd":20}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "jog"}

Description:

(a) The command sent means the robot joint 2 moves to 20 deg in the positive direction at 30 deg/s in joint space.

4.7 The joint moves to the specified position

Send message:

{"cmdName":"joint_move","relFlag":0,"jointPosition":[0,90.5,90.5,0,90.5,0],"speed":20.5,"accel":20.5}

Receive message:

{"errorCode": "0", "errorMsg": "", "cmdName": "joint_move"}

Description:

(a) jointPosition: [j1,j2,j3,j4,j5,j6] is the angle value of each joint you fill in the unit is degrees, not radians. Note that the positive and negative direction of the movement is reflected by the positive and negative value of the jointPosition.

(b) speed: speed_val represents the speed of the joint in (°/S), and the user can fill in appropriate parameters

(c) accel: accel_val represents the acceleration of the joint in (°/S^2）.The user can enter appropriate parameters, and the acceleration is recommended to be less than 720.

(d) relFlag: 0 or 1, 0 for absolute movement and 1 for relative movement

Notes:

joint_move is a block move command that one move command will only be executed after the proceeding move command is completed. If the next command needs to be executed immediately, it is recommended to use stop_program to stop the current movement first, and then send the next move command.

4.8 The TCP end moves to the specified position

Send message:

{"cmdName":"end_move","endPosition":[x,y,z,a,b,c],"speed":speed_val, "accel":accel_val}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "end_move"}

Description:

(a) endPosition: [x,y,z,a,b,c] specifies the value of xyzabc of the TCP end

(b) speed: speed_val represents the speed of the joint in degrees/S, and the user can fill in the appropriate parameters. If speed_val is set to 20, it means the joint speed is 20°/S.

(c) accel: accel_val represents the acceleration of the joint in (°/S^2）.The user can enter appropriate parameters, and the acceleration is recommended to be less than 720.

(d) end_move is a block move command that one move command will only be executed after the proceeding move command is executed. If the next command needs to be executed immediately, it is recommended to use stop_program to stop the current movement first, and then send the next move command.

Notes:

(a) end_move command does not move from the current position to the target position point in a straight line. This command first performs the inverse solution to the target point of cartesian space input by the user, and then uses the joint_move command to make the robot joint move to the specified position.

(b) If you want to move from the current position to the target position point in a straight line, use the moveL command.

4.9 MoveL

Send message:

{"cmdName":"moveL","relFlag":1,"cartPosition":[0,0,50,0,0,0],"speed":20,"accel":50,"tol":0.5}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "moveL"}

Description:

(a) cartPosition: [x,y,z,rx,ry,rz] specifies the value of x, y, z, rx, ry, rz of TCP end

(b) speed: speed_val stands for the linear speed in mm/s, and the user can input appropriate parameters. If speed_val is set to 20, it means the linear speed is 20 mm/s.

(c) accel: accel_val stands for the Accel of linear move(mm/S^2). The user can enter appropriate parameters and the value of Accel is recommended not to exceed 8000.

(d) relFlag: the value of flag_val can be 0 or 1. 0 stands for absolute movement, 1 stands for relative movement.

(e) moveL is a block move command that one move command will only be executed after the proceeding move command is executed. If the next command needs to be executed immediately, it is recommended to use stop_program to stop the current movement first, and then send the next move command.

4.10 MoveC

Send message:

```
{
    "cmdName":"movc",
    "relFlag":move_mode,
    "pos_mid":[x,y,z,rx,ry,rz],
    "pos_end":[x,y,z,rx,ry,rz],
    "speed":vel,
    "accel":acc,
    "tol":tol
}
```

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "movc"}

Description:

(a) pos_mid: [x,y,z,rx,ry,rz] specifies the value of x,y,z,rx,ry,rz in Cartesian space.

(b) pos_end: [x,y,z,rx,ry,rz] specifies the value of x,y,z,rx,ry,rz in Cartesian space.

(c) relFlag: the value of flag_val can be 0 or 1. 0 stands for absolute movement, 1 stands for relative movement.

(d) speed: speed_val represents the speed of the joint in degrees/S, and the user can fill in the appropriate parameters. If speed_val is set to 20, it means the joint speed is 20°/S.

(e) accel: accel_val represents the acceleration of the joint in (°/S²). The user can enter appropriate parameters, and the acceleration is recommended to be less than 720.

(f) tol: tol stands for Maximum Permissible Error, or 0 if not required.

4.11 Quit connection

Send message: {"cmdName":"quit"}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "quit"}

4.12 Get robot status

Send message: {"cmdName":"get_robot_state"}

Receive message:

```
{
    "cmdName":"get_robot_state",
    "errorCode":"0",
    "errorMsg":"unknown",
    "enable":"robot_enabled",
    "power": "powered_on"
}
```

Description:

Robot status: robot_enabled/robot_disabled   powered_on/powered_off

(a) robot_enabled/robot_disabled: indicates the enabled/disabled status of the robot

(b) powered_on/powered_off: indicates the Poweron/off status of the robot

4.13 Check if in servo_move mode

Send message: {"cmdName":"is_in_servomove"}

Receive message:

{"errorCode": "0", "errorMsg": "", "in_servomove": true, "cmdName": "is_in_servomove"}

Description:

(a) relFlag: 1 stands for entering servo_move mode, 0 stands for quit

4.14 Servo move control mode

Send message: {"cmdName":"servo_move","relFlag":0}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "servo_move"}

Description:

(a) relFlag: 1 stands for entering servo_move mode, 0 stands for quit.

4.15 Joint move control mode

Send message: {"cmdName":"servo_j","relFlag":1,"jointPosition":[0.1,0,0,0,0,0],"stepNum":1}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "servo_j"}

Description:

(a) The user can get the angles of the six joints and send them to the robot, so as to control it move to the corresponding end position

(b) Before using the servo_j command, the user needs to use the servo_move command to enter the servo position control mode

(c) jointPosition:[joint1,joint2,joint3,joint4,joint5,joint6] The angle of each joint in degrees is filled in.

(d) relFlag: 0; relFlag has optional parameters of 0 and 1. 0 stands for absolute movement, 1 stands for relative movement.

(e) stepNum: cycle frequency division, The robot will execute the received command in a cycle of mm * 8ms;

(f) Note that since the control period of the controller is 8ms, this command needs to be sent every 8ms to be effective, and it needs to be sent continuously as the effect is not visible by sending it only once. Moreover, the maximum speed of six joints is 180 degrees/s.

For example, [1.5,0.5,0.5,0.5,0.5,0.5,0.5], 1.5/0.008=187.5 exceeds the joint speed limit, so the servo_j command will not take effect.

Notes: This command is quite different from the fifth command joint_move since it is mainly used for trajectory planning in scientific research, and when this command is sent to the robot, it is directly sent to the servo instead of interpolated by the planner of the controller. When using this command, the user need to plan the trajectory in advance, otherwise the effect will be poor and cannot meet the expectation. In general, joint_move command is recommended.

4.16 Cartesian move control mode

Send message: {"cmdName":"servo_p","catPosition":[x,y,z,a,b,c],"relFlag":0,"stepNum":1}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "servo_p"}

Description:

(a) The user can use his/her own trajectory planning algorithm to generate the point coordinates of each joint and send them to the robot's six joints, so that the robot can execute these commands.

(b) Before using the servo_j command, the user needs to use the servo_move command to enter the servo position control mode

(c) catPosition: [x,y,z,a,b,c] indicates the coordinates of the robot in cartesian space; xyz is in mm and abc is in degrees.

(d) abc stands for the RPY angles of the robot pose, in the order of XYZ

(e) relFlag: 0; relFlag has optional parameters of 0 and 1. 0 stands for absolute movement, 1 stands for relative movement.

(g) stepNum: cycle frequency division, The robot will execute the received command in a cycle of mm * 8ms;

(f) Note that this command needs to be sent every 8ms to be effective. Because the controller interpolates a position point every 8ms. Moreover, the values of x, y, z, a, c and c should not be overlarge to avoid exceeding the safe speed limit of the controller.

Notes: This command is mainly used for trajectory planning in scientific research. When this command is used, it is not interpolated by the controller planner. When using this command, the user need to plan the trajectory in advance, otherwise the effect will be poor and can not meet the expectation. In general, moveL is recommended.

4.17 Set robot rapid rate

Send message: {"cmdName":"rapid_rate","rate_value":0.2}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "rapid_rate"}

Description:

(a) rate_value: 0.2 By default, the rapid rate of the robot is 1, which can be modified through this command, ranging from 0 to 1.

4.18 Loading program

Send message: {"cmdName":"load_program","programName":"track/test.jks"}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "load_program"}

4.19 Get the name of the loaded program

Send message: {"cmdName":"get_loaded_program"}

Receive message:

```
{
    "errorCode": "0",
    "errorMsg": "",
    "cmdName": "get_loaded_program",
    "programName": "test.jks"
}
```

4.20 Play program

Send message: {"cmdName":"play_program"}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "play_program"}

4.21 Pause program

Send message: {"cmdName":"pause_program"}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "pause_program"}

4.22 Resume program

Send message: {"cmdName":"resume_program"}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "resume_program"}

4.23 Stop program

Send message: {"cmdName":"stop_program"}

Receive message:{"errorCode": "0", "errorMsg": "", "cmdName": "pause_program"}

Description:

(a) Stopping the program actually means stops the robot movement

(b) stop_program can be used to stop other robot move commands, such as joint_move/moveL. It is recommended to use stop_program when you need to control the robot to stop.

4.24 Get program status

Send message: {"cmdName":"get_program_state"}

Receive message:

{"errorCode": "0", "errorMsg": "", "cmdName": "get_program_state", "programState": "idle"}

Description:

(a) programState: running/paused/idle

4.25 Set the DO value

Send message: {"cmdName":"set_digital_output","type":0,"index":1,"value":1}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "set_digital_output"}

Description:

(a) "type" refers to DO Type: 0 stands for controller DO, 1 stands for tool DO, and 2 stands for extend DO

(b) "index" refers to the number of DO. For example, the number of controller DO is 1 ~ 31. If you want to control the 31st DO, set the index to 31.

(c) "value" refers to the DO value, optionally 0 or 1

4.26 Get DI status

Send message: {"cmdName":"get_digital_input_status"}

Receive message:

```
{
    "din_status": [
        1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
    ],
    "errorCode": "0",
    "cmdName": "get_digital_input_status",
    "errorMsg": ""
}
```

Description:

(a) din_status: there are 136 pins in total. The first 8 pins are hardware IO of the cabinet. The first pin is occupied by the controller and cannot be used by the user.

Notes: The rest 56 pins are modbus IO.

4.27 Set the AO value

Send message: {"cmdName":"set_analog_output","type":type,"index":index,"value":value}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "set_analog_output"}

Description:

(a) "type" refers to AO Type. 0 stands for controller AO, and 2 stands for extend AO

(b) "index" is the number of AO. For example, the number of controlled AO is 1 ~ 7. If you want to control the 7th AO, set the index to 7.

(c) "value" refers to the AO value. Enter a floating point to meet the programming requirements, such as 4.32.

4.28 Set tool coordinate frame

Send message: {"cmdName":"set_tool_offsets","tooloffset":[10,10,10,10,10,10],"id":2,"name":"tcp_new"}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "set_tool_offsets"}

Description:

(a) tooloffset: fill in the x, y, z, rx, ry, rz of the tool coordinate frame. Such as [10,10,10,10,10,10]

(b) id: the ID number of the tool coordinate frame you want to set, with optional IDs 1 through 10. For example, set the tool coordinate frame id to 2.

(c) name: the name of the tool coordinate frame you want to set. For example, set the name to tcp_new.

4.29 Get tool coordinate frame

Send message: {"cmdName":"get_tool_offsets"}

Receive message:

```
{
    "errorCode": "0",
    "errorMsg": "",
    "tool_offsets": [
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 10.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
    ],
    "cmdName": "get_tool_offsets"
}
```

Description:

(a) tool_offsets: tool coordinate frame id corresponds to array index(0 ~ 10)

4.30 Select tool coordinate frame

Send message: {"cmdName":"set_tool_id", "tool_id":2}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "set_tool_id"}

Description:

(a) tool_id: tool coordinate frame ID number. Used to select which tool coordinate frame the robot will move in. For example, if the ID is set to 2, the robot will move in the second tool coordinate frame.

Note: when tool_id is 0, the robot's tool is by default the center of the end flange.

4.31 Set user coordinate frame

Send message:

{"cmdName":"set_user_offsets", "userffset":[10,10,10,10,10,10], "id":2, "name":"user_new"}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "set_user_offsets"}

Description:

(a) userffset: Fill in the x, y, z, rx, ry, rz of the user coordinate frame, such as [10,10,10,10,10].

(b) id: the ID number of the user coordinate frame you want to set, with optional IDs 1 through 10. For example, set the user coordinate frame id to 2.

(c) name: the name of the user coordinate frame you want to set. For example, set the name to user_new.

4.32 Get user coordinate frame

Send message: {"cmdName":"get_user_offsets"}

Receive message:

```
{
    "user_offsets": [
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [-439.0, -112.0, 303.0, -439.0, -112.0, 303.0],
        [9.0, 9.0, 9.0, 9.0, 9.0, 9.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
        [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
    ],
    "errorCode": "0",
    "cmdName": "get_user_offsets",
    "errorMsg": ""
}
```

Description:

(a) user_offsets : user coordinate frame id corresponds to array index(0 ~ 10)

4.33 Select user coordinate frame

Send message: {"cmdName":"set_user_id","user_frame_id":2}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "set_user_id"}

Description:

(a) set_user_id: user coordinate frame ID number. Used to select which user coordinate frame the robot will move in. For example, if the ID is set to 2, the robot will move in the second user coordinate frame.

(b) note that when set_user_id is set to 0, the robot is moving in the base coordinate frame.

4.34 Get extension IO status

Send message: {"cmdName":"get_extio_status"}

Receive message:

When there is no extension module:

(0, [], {'state': 0})

When there is extension module:

```
{
    "errorCode": "0",
    "errorMsg": "",
    "cmdName": "get_extio_status",
    "extio_status": "(1,[
            {
                'setup': (1, 'mobusIoName', ('192.168.2.47', 8888), {'ai': (0, 0), 'do': (0, 0), 'ao': (0, 0), 'di': (0, 8)}),
                'pinmap': {'ai': 0, 'do': 0, 'ao': 0, 'di': 0}
            }
        ],
        {'state': 0, 'di': (0, 0, 0, 0, 0, 0, 0, 0)}
    )"
}
```

Description:

extio_status is described

(a) The digit 1 in red font means there is an external extension IO module.

(b) setup: (1, 'mobusIoName', ('192.168.2.47', 8888))

1 stands for TCP and 0 stands for RTU. modbusIoName stands for the name of the external extension module, and ('192.168.2.47', 8888) stands for the IP and port.

(c) {'ai': (0, 0), 'do': (0, 0), 'ao': (0, 0), 'di': (0, 8)} : The 1st digit in the brackets stands for the initial assigned address of the external extension module register. The 2nd digit stands for the number of IO.

(d) pinmap: the initial address of pin index of external module, which needs to be considered when there are multiple external modules.

(e) {'state': 0, 'di': (0, 0, 0, 0, 0, 0, 0, 0)})"}: state 0 means the extension module is not running, and 1 means the extension module is running.

4.35 Get the status of the function input pin

Send message: {"cmdName":"get_funcdi_status"}

Receive message:

```
{
    "errorCode": "0",
    "errorMsg": "",
    "funcdi_status": [[-1, -1], [-1, -1], [-1, -1], [-1, -1], [-1, -1], [-1, -1], [-1, -1], [-1, -1], [0,2], [-1,- 1], [-1, -1], [-1, -
        1]],
    "cmdName": "get_funcdi_status"
}
```

Description:

(a) funci_status: A total of 12 function input pins are returned, which are

(1) start program

(2) pause program

(3) resume program

(4) stop program

(5) power on

(6) power off

(7) enable robot

(8) disable robot

(9) level 1 reduced mode

(10) protective stop

(11) back to original position

(12) level 2 reduced mode

(b) [type,index]: [-1,-1] is the default value, indicating that no input pin is set as the function pin. There are three values for "type": 0 stands for the input pin of cabinet, 1 stands for the input pin of robot end tool, and 2 stands for the input pin of modbus. "index" stands for the index number of the input pin, indicating which pin is set as the function pin. The above [0,2] indicates the third pin of the cabinet (the index starts at 0) is set as the level 1 reduced mode function pin.

4.36 Torque control mode

Send message: {"cmdName":"torque_control_enable","enable_flag":1}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "torque_control_enable"}

Description:

(a) enable_flag:1 means entering the torque control mode (i.e. robot drag function), 0 means quit.

4.37 Get drag status

Send message: {"cmdName":"drag_status"}

Receive message: {"drag_status": "False", "errorCode": "0", "cmdName": "drag_status", "errorMsg": ""}

Description:

(a) when the value of drag status is 1, it means the robot is in drag mode, and users can perform drag programming on the robot. When the value is 0, it is not in the status.

4.38 Query user-defined system variables

Send message: {"cmdName":"query_user_defined_variable"}

Receive message:

```
{
    "errorCode": "0",
    "errorMsg": "",
    "var_list": [{"alias": " system_var1", "id": 5500, "value": 12.0}] ",
    cmdName": "query_user_defined_variable"
}
```

Description:

(a) var_list: all user-defined system variables will be returned in a list

(b) alias: the name of the system variable id: the id of the system variable. The user needs to know the id of the system variable if he/she wants to modify the system variables.

(c) value: refers to the value of the system variable. For example, in the message returned, the system variable name returned is system_var1, id is 5500, and value is 12.0

4.39 Modify user-defined system variables

Send message:

{"cmdName":"modify_user_defined_variable","id_new":5500,"alias_new ":"s_new","value_new":14}

Receive message:

{"errorCode": "0", "errorMsg": "", "cmdName": "modify_user_defined_variable"}

Description:

(a) id_new: refers to the id number of the system variable to be modified, which can be queried through the command query_user_defined_variable

(b) alias_new: sets the new name of the system variable to be modified

(c) value_new: sets the value of the system variable to be modified

4.40 Protective stop status

Send message: {"cmdName":"protective_stop_status"}

Receive message:

{"errorCode":"0", "errorMsg":"", "protective_stop":"0", "cmdName":"protective_stop_status"}

Description:

(a) protective_stop: When the value of protective_stop is 1, it means the robot is in protective stop status, and when the value is 0, it is not in this status. Protective stop usually occurs in the event of a collision between the robots. In this case, the robot enters the protective stop status.

4.41 wait_complete command

Send message: {"cmdName":"wait_complete"}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": " wait_complete"}

Notes: This command is deprecated and is not recommended.

4.42 Set payload

Send message: {"cmdName":"set_payload","mass":m,"centroid":[x,y,z]}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": " set_payload "}

Description:

(a) "mass" refers to the mass of payload, unit: KG.

(b) "centroid" refers to the centroid of payload, unit: mm.

For example, the payload mass to be set is 1KG, and the centroid is [10,10,10].Then you can send the command:

{"cmdName":"set_payload","mass":1,"centroid":[10,10,10]}

4.43 Get payload

Send message: {"cmdName":"get_payload"}

Receive message:

{"errorCode": "0", "centroid": [0.0, 0.0, 1.0], "mass": 1.0, "cmdName": "get_payload", "errorMsg": ""}

Description:

(a) "mass" refers to the mass of payload, unit: KG.

(b) "centroid" refers to the centroid of payload, unit: mm.

4.44 Set robot collision level

Send message: {"cmdName":"set_clsn_sensitivity","sensitivityVal":level}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "set_clsn_sensitivity"}

Description:

(a) The value range for "level" is 0 to 5.

0 means close collision protection.

Level 1 has the highest collision sensitivity.

Level 5 has the lowest collision sensitivity.

4.45 Get robot collision level

Send message: {"cmdName":"get_clsn_sensitivity"}

Receive message:

{"errorCode": "0", "sensitivityLevel": level, "cmdName": "get_clsn_sensitivity", "errorMsg": ""}

Description:

(a) "sensitivityLevel" refers to the collision level returned

0 means close collision protection.

1 stands for the highest collision sensitivity.

5 stands for the lowest collision sensitivity.

4.46 Robots forward kinematics solution

Send message: {"cmdName":"kine_forward","jointPosition":[j1,j2,j3,j4,j5,j6]}

Receive message:

```
{
    "errorCode": "0",
    "errorMsg": "",
    "cmdName": "kine_forward",
    "cartPosition": [x,y,z,rx,ry,rz]
}
```

Description:

(a) [j1,j2,j3,j4,j5,j6] refers to the six joint angles for forward kinematics solution,

(b) [x,y,z,rx,ry,rz] refers to the end pose of the robot obtained from forward kinematics solution.

4.47 Robots inverse kinematics solution

Send message: {"cmdName":"kine_inverse","jointPosition":[j1,j2,j3,j4,j5,j6],"cartPosition": [x,y,z,rx,ry,rz]}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "kine_inverse", "jointPosition":[x,y,z,rx,ry,rz]}

If inverse kinematics fails:

{"errorCode": "2", "errorMsg": "call kine_inverse failed", "cmdName": "kine_inverse"}

Description:

(a) The jointPosition sent is the reference joint angle of the robot. It is recommended to select the current joint angle of the robot as the reference joint angle. The unit is degrees.

(b) The cartPosition in the sent message is the end pose of the robot. [x,y,z,rx,ry,rz] xyz is in millimeters and rx, ry, rz is in degrees.

(c) The jointPosition obtained in the received message refers to the joint angle through inverse kinematics solution.

4.48 Recover from Collision Protection mode after collision

Send message: {"cmdName":"clear_error"}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "clear_error"}

4.49 Get joint position

Send message: {"cmdName":"get_joint_pos"}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "get_joint_pos", "joint_pos": [j1,j2,j3,j4,j5,j6]}

Description:j1, j2, j3, j4, j5, j6 refer to the positions of the six returned joints, in degrees.

4.50 Get the end pose in the current tool coordinate frame

Send message: {"cmdName":"get_tcp_pos"}

Receive message:

{"errorCode": "0", "errorMsg": "", "cmdName": " get_tcp_pos", "tcp_pos": [x,y,z,a,b,c]}

Description:[x,y,z,a,b,c] refers to the end pose in the currently set tool coordinate frame which, by default, is the center of the flange.

4.51 Get controller version

Send message: {"cmdName": "get_version"}

Receive message: {"errorCode": "0", "errorMsg":"", "version": "1.6.5_01_X86", "cmdName": "get_version"}

4.52 Set trajectory capture parameters

Send message: {"cmdName": "set_traj_config", "acc":100, "vel":20, "xyz_interval":0.1, "rpy_interval":0.1}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_traj_config"}

Description:

(a) acc: acceleration of trajectory execution mm/s^2.

(b) ve: velocity of trajectory execution mm/s.

(c) xyz_interval: time intervals of displacement data acquisition in Cartesian space.

(d) rpy_interval: time intervals of rotary data acquisition in Cartesian space.

4.53 Get trajectory capture parameters

Send message: {"cmdName": "get_traj_config"}

Receive message:

```
{
    "acc":100.0,
    "xyz_interval": 0.1,
    "errorMsg": "",
    "cmdName": "get_traj_config",
    "errorCode": "0",
    "rpy_interval": 0.1,
```

"vel":20.0

}

Description:

(a) acc: acceleration of trajectory execution mm/s^2.

(b) ve: velocity of trajectory execution mm/s.

(c) xyz_interval: time intervals of displacement data acquisition in Cartesian space.

(d) rpy_interval: time intervals of rotary data acquisition in Cartesian space.

4.54 Set trajectory capture switch

Send message: {"cmdName": "set_traj_sample_mode", "mode": 1, "filename": "test"}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "set_traj_sample_mode"}

Description:

(a) mode:0 off, 1 on.

(b) filename: filename.

4.55 Get trajectory capture switch status

Send message: {"cmdName": "get_traj_sample_status"}

Receive message:

{"sampleStatus": 1, "errorCode": "0", "cmdName": "get_traj_sample_status", "errorMsg": ""}

Description:

(a) sampleStatus:0 off, 1 on.

4.56 Get existing trajectory file name

Send message: {"cmdName": "get_exist_traj_file_name"}

Receive message:

```
{
    "errorCode": "0",
    "errorMsg": "",
    "cmdName": "get_exist_traj_file_name",
    "trajTrackFileName": ["pythonTrack1", "test"]
}
```

Description:

(a) trajTrackFileName: Track data filename.

4.57 Rename existing trajectory file

Send message: {"cmdName": "rename_traj_file_name", "src": "test", "dest": "test_rename"}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "rename_traj_file_name"}

Description:

(a) src: Source trajectory data filename

(b) dest: Name of the filename modified.

4.58 Remove trajectory

Send message: {"cmdName": "remove_traj_file", "fileName": fileName}

Receive message: {"errorCode": "0", "errorMsg": "", "cmdName": "remove_traj_file"}

Description:

(a) filename: trajectory data filename

4.59 Generate the trajectory execution script

Send message: {"cmdName": "generate_traj_exe_file", "fileName": "test_rename"}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "generate_traj_exe_file"}

Description:

(a) filename: trajectory data filename

4.60 Servo filter

The servo filter is set by set_servo_move_filter instruction to assist in planning trajectories

filter_type stands for move mode, and there are 6 optional values:

no_filter disable filter: 0

First order low-pass filter in joint space: 1

Nonlinear filter in joint space: 2

Multi-order mean filter in joint space: 3

Nonlinear filter in Cartesian space: 4

Speed forward: 5

When filter_type is 0, there is only 1 parameter

Send message: {"cmdName": "set_servo_move_filter", "filter_type":0}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_servo_move_filter"}

Description:

Disable filter. The filter is not used for auxiliary planning, which can be set only after exiting the servo mode. .

When filter_type is 1, there are only 2 parameters

Send message: {"cmdName": "set_servo_move_filter", "filter_type":1",lpf_cf":0.5}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_servo_move_filter"}

Description:

(a) Setting of first order low-pass filter in joint space can be done only after exiting the servo mode.

(b) lpf_cf stands for cutoff frequency in HZ.

When filter_type is 2, there are only 4 parameters

Send message: {"cmdName": "set_servo_move_filter", "filter_type":2, "nlf_max_vr":2, "nlf_max_ar":2, "nlf_max_jr":4}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_servo_move_filter"}

Description:

(a) Setting of nonlinear filter in joint space can be done only after exiting servo mode.

(b) max_vr The upper limit of change speed (absolute value) °/s

(c) max_ar The upper limit of accelerated speed of change speed (absolute value) °/s^2

(d) max_jr The upper limit value of jerk (absolute value) of change speed °/s^3

When filter_type is 3, there are only 5 parameters

Send message:

```
{
    "cmdName": "set_servo_move_filter",
    "filter_type":3,
    "mmf_max_buf":20,
    "mmf_kp":0.1,
    "mmf_kv":0.2,
    "mmf_ka":0.6
}
```

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_servo_move_filter"}

Description:

(a) Setting of multi-order mean filter in joint space can be done only after exiting the servo mode.

(b) mmf_max_buf: The wider the filter window, the smoother the filter trajectory, which will cause the the loss of precision

(c) mmf_kp: Position tracking coefficient. The smaller it is, the smoother the filter trajectory will be. The larger it is the higher the precision will be. But there may be jitter.

(d) mmf_kv: The smaller the speed tracking coefficient the smoother the filter trajectory will be, and the larger it is the higher the precision will be, but there may be jitter.

(e) mmf_ka: The smaller the acceleration position tracking coefficient the smoother the filter trajectory, and the larger it is the higher the precision will be, but there may be jitter.

When the value of filter_type is 4, there are only 7 parameters

Send message:

```
{
    "cmdName": "set_servo_move_filter",
    "filter_type":4,
    "nlf_max_vr":2,
    "nlf_max_ar":2,
    "nlf_max_jr":4,
    "nlf_max_vp":10,
    "nlf_max_ap": 100,
    "nlf_max_jp":200
}
```

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_servo_move_filter"}

Description:

(a) Setting of the nonlinear filter in Cartesian space can be done only after exiting the servo mode.

(b) max_vr The upper limit of Cartesian space orientation change speed (absolute value) °/s

(c) max_ar The upper limit of accelerated speed of Cartesian space orientation change speed (absolute value)°/s^2

(d) max_jr The upper limit value of jerk (absolute value) of Cartesian space orientation change speed °/s^3

(e) max_vp The upper limit (absolute value) of the move command speed in Cartesian space. Unit: mm/s

(f) max_ap The upper limit (absolute value) of the move command accelerated speed in Cartesian space. Unit: mm/s^2

(g) max_jp The unit of upper limit (absolute value) of the move command jerk in Cartesian space. mm/s^3

When the value of filter_type is 5, there are only 3 parameters

Send message:

{"cmdName": "set_servo_move_filter", "filter_type":5, "mmf_max_buf":max_buf, "mmf_kp":kp}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_servo_move_filter"}

Description:

(a) Setting of multi-order mean filter in joint space can be done only after exiting the servo mode.

(b) max_buf The buffer size of the mean filter.

(c) mmf_kp Acceleration filter factor

4.61 Set sensor brand

Send message: {"cmdName": "set_torsenosr_brand", "sensor_brand":num}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_torsenosr_brand"}

Description:

(a) sensor_band:1 is SONY Semiconductor; 2 is BoschSensortec; 3 is STMicroelectronics

4.62 Get sensor brand

Send message: {"cmdName": "get_torsenosr_brand"}

Receive message:{"sensor_brand":1, "errorCode": "0", "cmdName": "get_torsenosr_brand", "errorMsg":""}

Description:

(a) sensor_band:1 is SONY Semiconductor; 2 is BoschSensortec; 3 is STMicroelectronics

4.63 Set compliance control parameters

Send message:

```
{
    "cmdName": "set_admit_ctrl_config",
    "axis":2,
    "opt":1,
    "ftUser":25,
    "ftConstant":5,
    "ftNormalTrack":0,
    "ftReboundFK":0
}
```

Receive message:{"errorCode": "0", "errorMsg":"", "cmdName": "set_admit_ctrl_config"}

Description:

(a) axis:0:x-axis; 1: y-axis; 2: z-axis; 3: rx4: ry5: rz

(b) opt: 0 off 1 on

(c) 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.

(d) ftReboundFK: The rebound force, indicating the ability of the robot to return to its initial state.

(e) ftConstant: The constant force, all of which are set to 0 when manual operation.

(f) ftNormalTrack: normal tracking, all of which are set to 0 when manual operation.

4.64 Get the force-control compliance control parameters

Send message: {"cmdName": "get_admit_ctrl_config"}

Receive message:

```
{
    "errorCode": "0",
    "errorMsg":"",
    "cmdName":"get_admit_ctrl_config",
    "admitConfig":[
        [0,5.0,0.0,0.0,0],
        [0,5.0,0.0,0],
        [1,25.0,0.0,5.0,0],
        [0,0.20000000298023224,0.0,0.0,0],
        [0,0.20000000298023224,0.0,0.0,0],
        [0,0.20000000298023224,0.0,0.0,0]
    ]
}
```

Description:

(a) admitConfig: There are 6 groups of 1*5 matrices representing the force control and compliance control parameters of x, y, z, rx, ry, rz respectively.

(b) Within each matrix in order [opt,ftUser,ftReboundFK,ftConstant,ftNnormalTrack].

4.65 Set the F/T sensor switch

Send message: {"cmdName": "set_torque_sensor_mode", "sensor_mode":mode}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_torque_sensor_mode"}

Description:

(a) mode: 0 on 1 off

4.66 Start to identify the tool end payload

Send message: {"cmdName": "start_torq_sensor_payload_identify", "jointPosition":[j1,j2,j3,j4,j5,j6]}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "start_torq_sensor_payload_identify"}

Description:

(a) This is a movement instruction that moves to the position specified by jointposition.

(b) jointPosition: The identification payload end point: it is recommended that the 456 axis be rotated 90 degrees to obtain accurate identification results.

4.67 Get end payload identification status

Send message: {"cmdName": "get_torq_sensor_identify_staus"}

Receive message:

{"errorCode": "0", "errorMsg":"", "cmdName": "get_torq_sensor_identify_staus", "identify_status":1}

Description:

(a) identify_status: 0 means identification completed, 1 means identification uncompleted, 2 means identification failed.

4.68 Get end payload identification result

Send message: {"cmdName": "get_torq_sensor_payload_identify_result"}

Receive message:

```
{
    "errorCode": "0",
    "centroid":[0,0,0],
    "mass":0,
    "cmdName":
    "get_torq_sensor_payload_identify_result",
    "errorMsg":""
}
```

Description:

(a) "mass" refers to the mass of payload, unit: KG.

(b) "centroid" refers to the centroid of payload, unit: mm.

4.69 Set sensor end load

Send message: {"cmdName": "set_tool_payload", "mass":weight, "centroid":[x,y,z]}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_tool_payload"}

Description:

(a) "mass" refers to the mass of payload

(b) unit: KG. "centroid" refers to the centroid of payload, unit: mm.

For example, the payload mass to be set is 1KG, and the centroid is [10,10,10]. Then you can send the command:

```
{"cmdName":"set_payload","mass":1,"centroid":[10,10,10]}
```

4.70 Get sensor end load

Send message: {"cmdName": "get_tool_payload"}

Receive message:

{"errorCode": "0", "centroid":[0.0,0.0,0.0], "mass":1.0, "cmdName": "get_tool_payload", "errorMsg":""}

Description:

(a) "mass" refers to the mass of payload, unit: KG.

(b) "centroid" refers to the centroid of payload, unit: mm.

4.71 set admittance control switch

Send message: {"cmdName": "enable_admittance_ctrl", "enable_flag":1}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "enable_admittance_ctrl"}

Description:

(a) enable_flag: 0 off 1 on.

4.72 Get the force control type and sensor initial state

Send message: {"cmdName": "get_compliant_type"}

Receive message:

```
{
    "errorCode": "0",
    "errorMsg":"",
    "compliance_type":0,
    "sensor_compensation":1,
    "cmdName": "get_compliant_type"
}
```

Description:

(a) sensor_compensation: Turn on sensor compensation enable flag. 1 means turning on initialization, and 0 means no initialization

(b) compliance_type: 0 means not using any kind of compliance control method, 1 means using constant compliance control, and 2 means using speed compliance control

4.73 Set the force control type and sensor initial state

Send message: {"cmdName": "set_compliant_type", "sensor_compensation":1, "compliance_type":1}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_compliant_type"}

Description:

(a) sensor_compensation: Turn on sensor compensation enable flag. 1 means turning on initialization, and 0 means no initialization

(b) 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

4.74 Set the compliance control torque conditions

Send message: {"cmdName": "set_compliance_condition", "compliance_condition":[0,0,0,0,0,0]}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_compliance_condition"}

Description:

(a) compliant_condition: force component and moment component of the force sensor, fx: force component along x-axis; tx: moment component around x-axis

4.75 Set network anomaly conditions

Send message: {"cmdName": "set_network_exception_handle", "timeLimit":100, "motType":0}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_network_exception_handle"}

Description:

(a) millisecond time parameter, in milliseconds, how many ms delay will it trigger upon network anomaly

(b) mnt: the motion types the robot needs to perform upon network anomaly. 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.

4.76 Set the F/T sensor IP address

Send message:

{"cmdName": "set_torque_sensor_comm", "type":0, "ip_addr": "172.30.1.110", "port":55555}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_torque_sensor_comm"}

Description:

(a) type: 0 is using tcp/ip protocol, 1 is using RS485 protocol

(b) ip_addr is the address of the force control sensor

(c) port is the port number of force control sensor when using tcp/ip protocol

4.77 Get the F/T sensor IP address

Send message: {"cmdName": "get_torque_sensor_comm"}

Receive message:

```
{
    "ip_addr": "192.168.2.228",
    "errorMsg":"",
    "cmdName": "get_torque_sensor_comm",
    "errorCode": "0",
    "type":0,
    "port":49152
}
```

Description:

(a) type: 0 is using tcp/ip protocol, 1 is using RS485 protocol

(b) ip_addr: The force control sensor address

(c) port: port number of the force control sensor when using tcp/ip protocol

4.78 Disable the torque control

Send message: {"cmdName": "disable_force_control"}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "disable_force_control"}

4.79 Set speed compliance control parameters

Send message:

{"cmdName": "set_vel_compliant_ctrl", "compliant_ctrl":[vc_level,rate1,rate2,rate3,rate4]}

Receive message: {"errorCode": "0", "errorMsg":"", "cmdName": "set_vel_compliant_ctrl"}

Description:

(a) compliant_ctrl:

vc_level speed compliance control level

rate1: Rate level 1

rate2: Rate level 2

rate3: Rate level 3

rate4: Rate level 4

4.80 Setting of tioV3 tool voltage

Send message: {"cmdName": "set_tio_vout_param", "tio_vout_ena":0, "tio_vout_vol":0}

Receive message:

{"errorCode": "0", "errorMsg":"", "cmdName": "set_tio_vout_param"}

Description:

(a) tio_vout_ena parameter is 0 or 1; 0 means enabled, 1 means unenabled

(b) tio_vout_vol parameter is 0 or 1; 0 is 12V, 1 is 24V

4.81 Get tioV3 tool voltage

Send message:

{"cmdName":"get_tio_vout_param"}

Receive message:

{"errorCode": "0", "errorMsg":"", "tio_vout_ena":0, "tio_vout_vol":0, "cmdName": "get_tio_vout_param"}

Description:

(a) tio_vout_ena parameter is 0 or 1; 0 means enabled, 1 means unenabled

(b) tio_vout_vol parameter is 0 or 1; 0 is 12V, 1 is 24V

4.82 Get robot DH parameters

Send message: {"cmdName":"get_dh_param"}

Receive message: {"errorCode": "0", "errorMsg": "", "alpha": [0,0,0,0,0,0], "a": [0,0,0,0,0,0], "d": [0,0,0,0,0,0],"joint_homeoff": [0,0,0,0,0,0], "cmdName": "get_dh_param"}

Chapter 5 Error Description

errorCode	errorMsg	Description
0	The message returned is empty	Message executed successfully
1	Exception: function call failed	An exception occurred in the program
2	Return error message	Error Message

Chapter 6 Introducing JSON

JSON(JavaScript Object Notation) is a lightweight data-interchange format. Simply, JSON could transfer the JavaScript Object into text format, afterwards, it is easy to transmit this text between functions or asynchronous application like sending some data from web client to server. It's easy to interpret although weird. Further, object more complex than name/value could be described by JSON.

Chapter 7 Reference Code

Please refer to the attached Python code, and there are detailed descriptions on key contents in the code. The Python file is the demo used by the TCP command. When using the file, you need to change the IP address in the file to the IP address of your robot. The port number is static and does not need to be modified.

The JSON syntax is subset of the JavaScript syntax.

Data is written as name/value pairs. Objects are delimited with curly brackets and use comma to separate each pair. JSON basic types data types are: Number (integer or floating-point), String (delimited with the double quotation marks), Boolean (either of the values true or false), Array (delimited with the square brackets), Object(delimited with curly brackets).

Chapter 8 Feedback and Reports

Please send us a report if there are any inaccurate descriptions or errors in the documentation. You can also send an email to support@jaka.com if your have any ideas or comments. Then, our colleagues will try their best to reply one by one.

JAKA ROBOTICS CO., LTD.

Add.: Building 33-35, 610, Jianchuan Rd., Minhang District, Shanghai

E-mail: support@jaka.com

Web: www.jakarobotics.com