JAKA®

节卡机器人

TCP 协议

翻译版本（zh）

文档版本：2.0.4

注意：

协作机器人的定义遵循国际 ISO 标准及国标相关规定来保护作业者的安全，我们不推荐直接将机器人本体应用于作业对象为人体的场合。但机器人应用者或应用开发者确有需要涉及机器人作业对象为人体的场合时，需要在应用者或应用开发者充分评估并在人员安全得到保障的前提下，为机器人本体配置安全可靠、经过充分测试及认证的安全防护系统，以保护人员安全。

节卡会定期对手册进行修正和完善，其内容可能会更新，恕不另行通知。使用本手册前请认真核对实际产品信息。

本手册适用于节卡推出的全部产品和/或服务（后文统称为“产品”），手册所包含的信息按产品“现状”提供，受中华人民共和国法律（不包括香港、澳门与台湾地区法律）的约束并依据其解释，在法律允许的最大范围内，本手册不构成节卡任何形式的明示或暗示的陈述或保证，亦不构成对节卡有关产品的适销性、适用于特定目的、达到预期效果以及不侵权的保证。节卡对本手册中仍然可能出现的任何错误、遗漏以及对使用本手册及其所介绍产品而引起的意外或间接伤害概不负责。安装、使用产品前，请仔细阅读本手册。

本手册图片仅供参考，请以实物为准。

若节卡机器人产品出现被改造或者拆卸的情况，节卡不负责无偿售后工作。

节卡提醒用户在使用、维修节卡机器人时必须使用安全设备，必须遵守安全条款。

节卡机器人的程序设计者、机器人系统的设计和调试者，必须熟悉节卡机器人的编程方式和系统应用安装。

手册使用说明

本手册主要内容为节卡机器人 TCP 协议下的接口说明、接口命令格式等。

本手册面向的用户应接受过基本的编程培训，这将更加有助于机器人的使用。

更多信息

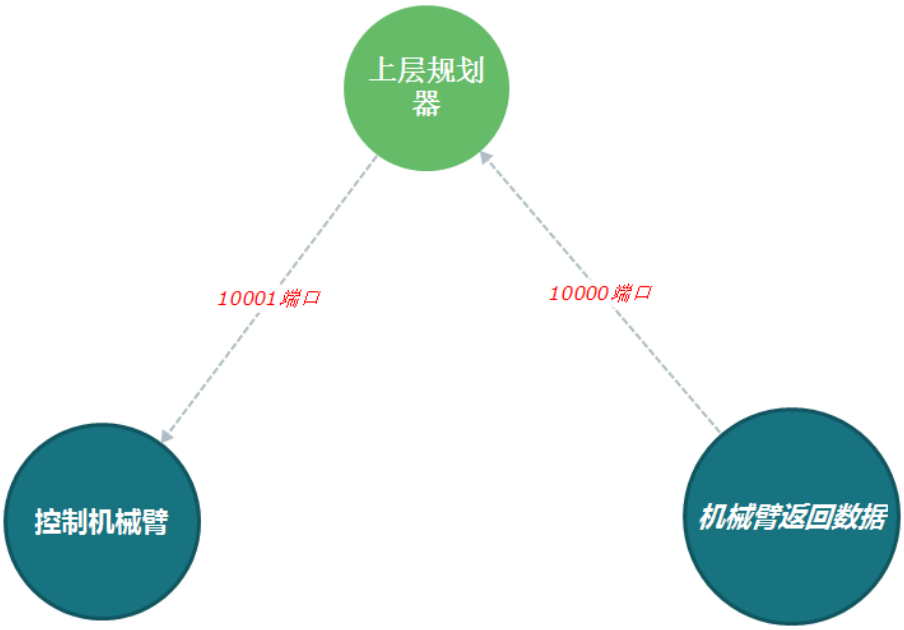
如果您还想了解更多的产品信息，请扫描右侧二维码访问我们的官网 www.jaka.com。

目录

第 1 章 用户开发说明

用户需要根据 JAKA TCP 协议进行机器人二次开发。

* 控制器启动一个 tcp 服务端，用于接收用户自定义的指令；
* 用户需要根据本协议实现相应的 tcp 客户端调用，即用户需要自己设计上层规划器发送机器人数据；
* 该功能可以与 JAKA-APP 同时使用，不需要断开 APP 与控制器的连接；
* 服务器监听端口号：10001 用来给机器人发送控制指令；
* 服务器监听端口号：10000 用来接收机器人返回的数据；
* 使用本协议进行二次开发难度较大，无特殊需求，建议使用本公司的 JAKA SDK 进行二次开发。



第 2 章 10000 端口说明

10000 端口被设计来返回机器人的数据，和机器人有关的数据以字符串的形式返回，返回的样例内容如下：


```
{
    "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
}
```


说明：

1. len：数据包的长度，一般最大长度不超过 5000 字节。
2. drag_status：布尔值，表示是否处于拖动模式。
3. extio：扩展 IO 模块的配置和状态，内部数据结构如下。

(1) status：所有扩展 IO 引脚的状态，包括 DI、DO、AI、AO。

status[0] 所有扩展 DI 的状态，

status[1] 所有扩展 DO 的状态，

status[2] 所有扩展 AI 的状态，

status[3] 所有扩展 AO 的状态。

(2) setup：所有扩展 IO 配置的数组，最多 8 个。对于每个扩展 IO 模块，设置遵循以下结构。

setup[N][0]：通讯类型，0：Modbus RTU，1：ModbusTCP/IP。

setup[N][1]：扩展 IO 模块的别名。

setup[N][2]：通信配置，根据通信类型不同。

① 对于 modbus RTU（设置[N][0] == 0）：

setup[N][2][0]：波特率

setup[N][2][1]:数据位长度

setup[N][2][2]：从站号

setup[N][2][3]：数据位

setup[N][2][4]:奇偶校验位

setup[N][2][5]:停止位长度

② 对于 modbus TCP (setup[N][0] == 1)：

setup[N][2][0]：IP 地址

setup[N][2][1]：端口

setup[N][2][2]：从站号

(3) setup[N][3]：引脚配置

setup[N][3][0][0]：DI 中寄存器的起始偏移量

setup[N][3][0][1]:DI 个数

setup[N][3][1][0]：DO 中寄存器的起始偏移量

setup[N][3][1][1]:DO 数量

setup[N][3][2][0]:AI 中寄存器的起始偏移量

setup[N][3][2][1]:AI 数量

setup[N][3][3][0]：AO 中寄存器的起始偏移量

setup[N][3][3][1]:AO 数

(4) num：扩展 IO 模块个数

(5) pinmap：扩展 IO 模块的管脚图数组

pinmap[N][0]：设置[N]的 DI 中寄存器的起始偏移量

pinmap[N][1]：setup[N]的 DO 中寄存器的起始偏移量

pinmap[N][2]：setup[N]的 AI 中寄存器的起始偏移量

pinmap[N][2]：setup[N]的 AO 中寄存器的起始偏移量

(6) mode：扩展 IO 模块的状态，0 表示状态开启，1 表示关闭。

4. errcode：来自控制器的十六进制字符串错误代码，如“0x0”。
5. errmsg：来自控制器的错误信息字符串，对应 errcode。
6. monitor_data：用于诊断的机器人和机柜的监控数据。

monitor_data[0]：SCB 主要版本号

monitor_data[1]：SCB 次要版本号

monitor_data[2]：控制柜温度
monitor_data[3]：控制柜母线平均功率
monitor_data[4]：控制柜母线平均电流
monitor_data[5]：数组中机器人 1-6 关节轴的监控数据
monitor_data[5][0]：机器人轴的瞬时电流，
monitor_data[5][1]：机器人轴的瞬时电压，
monitor_data[5][2]：机器人轴的瞬时温度，
monitor_data[5][3]：瞬时平均功率，
monitor_data[5][4]：电流波动，
monitor_data[5][5]：累计跑圈，
monitor_data[5][6]：累计运行时间，
monitor_data[5][7]：本次开机后运行循环，
monitor_data[5][8]：本次开机后的运行时间，
monitor_data[5][9]：关节扭矩

7. torqsensor：扭矩设置和状态数组，数据结构如下。
   (1) torqsensor[0]：当前扭矩传感器的设置
      ① torqsensor[0][0]：通讯类型，0 为 TCP/IP 通讯，1 为串口通讯；
      ② torqsensor[0][1]：扭矩传感器的通讯配置。与 TCP/IP 地址通信时为 [ip, port]，与串口通信时为 [baudrate, databits, stopbits, parity]。
      ③ torqsensor[0][2]：扭矩传感器末端的有效载荷质量，以及有效载荷的中心位置。
   (2) torqsensor[1]：扭矩传感器的状态
      ① torqsensor[1][0]：扭矩传感器状态，1 表示工作，0 表示关闭。
      ② torqsensor[1][1]：扭矩传感器错误代码。
      ③ torqsensor[1][2]：扭矩传感器实际接触力（6 维）。
      ④ torqsensor[1][3]：扭矩传感器原始读数值（6 维）。
8. joint_actual_position：所有 6 个关节的实际位置。
9. actual_position：机器人 TCP 在笛卡尔空间中的实际位置。
10. din：数字输入数组的所有值的数组
11. dout：机柜中所有数字输出值的数组。
12. ain：柜内所有模拟输入值的数组。
13. aout：柜内所有模拟输出值的数组。
14. tio_din：所有工具数字输入的状态数组。
15. tio_dout：所有工具数字输出的状态数组。
16. tio_ain：所有工具模拟输入的状态数组。
17. task_state：电源状态和使能状态指示灯，1：机器人断电，2：机器人上电，3：机器人使能，4：机器人使能。
18. homed：每个关节的 home 状态（已废弃）
19. task_mode：机器人的任务模式。 1：手动模式，2：自动模式，3：保留模式，4：拖动模式。
20. interp_state：程序状态，0：空闲，1：加载，2：暂停，3：运行。
21. enabled：显示机器人启用状态的布尔值。
22. paused：显示程序暂停状态的布尔值。
23. rapidrate：机器人运动的缩放速率。
24. current_tool_id：当前 TCP 的索引。
25. current_user_id：当前用户坐标系的索引。

第 3 章 命令接口格式

消息格式：所有的消息内容均为 json 字符串格式；

* 消息发送格式：

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

* 消息接收格式：

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

第 4 章 可使用接口

4.1 打开电源

发送消息：{"cmdName":"power_on"}
接收消息： {"errorCode": "0", "errorMsg": "", "power status": "True", "cmdName": "power_on"}

4.2 关闭电源

发送消息：{"cmdName":"power_off"}
接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "power_off"}

4.3 机器人上使能

发送消息： {"cmdName":"enable_robot"}
接收消息： {"errorCode": "0", "enabled status": "True", "cmdName": "enable_robot", "errorMsg": ""}

4.4 机器人下使能

发送消息： {"cmdName":"disable_robot"}
接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "disable_robot"}

4.5 关闭机器人和控制器

发送消息： {"cmdName":"shutdown"}
接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "shutdown"}

4.6 Jog 指令

Jog 指令包括关节空间的单轴运动和笛卡尔空间的**单轴运动**
jog_mode 代表运动模式可填的值有 3 个:
Jog_stop(Jog 停止): 0
Continue(连续运动): 1
Increment(步进运动): 2
ABS 绝对运动: 3
coord_map 代表坐标系的选择可填的值有 3 个:
在世界坐标系下运动: 0
在关节空间运动: 1
在工具坐标系下运动: 2
jnum 在关节空间下代表轴号 1 轴到六轴的轴号分别对应数字 0 到 5
jnum 在笛卡尔空间代表 x，y，z，rx，ry，rz 分别对应数字 0 到 5
jogvel 代表速度
poscmd 代表步进值，单关节运动为 deg,空间单轴运动为 mm

当 **jog_mode 为 0** 的时候只有 3 个参数
发送消息： {"cmdName":"jog","jog_mode":0, "coord_map":1, "jnum":1}

接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "jog"}
说明：
(a) 发送的指令代表机器人停止机器人关节 2 在关节空间下的 jog 运动

当 **jog_mode 为 1** 的时候只有 4 个参数
发送消息： {"cmdName":"jog","jog_mode":1, "coord_map":1, "jnum":1, "jogvel":30}
接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "jog"}
说明：
(a) 发送的指令代表机器人在关节空间下，关节 2 以 30 deg/s 做 jog 运动
(b) 值的注意的是,运动的正负由 jogvel 的正负来确定
(c) 需要先停止当前轴的运动才能再次下发该轴的运动指令
当 **jog_mode 为 2** 的时候只有 5 个参数
发送消息： {"cmdName":"jog","jog_mode":2, "coord_map":1, "jnum":3, "jogvel":30, "poscmd":20}
接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "jog"}
说明：
(a) 发送的指令代表机器人关节 2 在关节空间下以 30 deg/s，向正方向运动了 20 deg。
值的注意的是，运动的正负可以由 poscmd 的正负来确定。

当 **jog_mode 为 3** 的时候只有 5 个参数
发送消息： {"cmdName":"jog","jog_mode":3, "coord_map":1, "jnum":3, "jogvel":30, "poscmd":20}
接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "jog"}
说明：
(b) 发送的指令代表机器人关节 2 在关节空间下以 30 deg/s，运动到 20 deg。

4.7 关节运动到指定位置 MoveJ

发送消息：
{"cmdName":"joint_move","relFlag":0,"jointPosition":[0,90.5,90.5,0,90.5,0],"speed":20.5,"accel":20.5}
接收消息：
{"errorCode": "0", "errorMsg": "", "cmdName": "joint_move"}
说明：
(a) relFlag: 可选值为 0 或者 1, 0 代表绝对运动，1 代表相对运动
(b) jointPosition：[j1,j2,j3,j4,j5,j6] 填入的是每一个关节的角度值
单位是度，不是弧度。值的注意的是运动的正负由 jointPosition 值的正负来确定。
(c) speed: speed_val 代表关节的速度，单位是 (°/S)，用户可以自行填入适合的参数
(d) accel：accel_val 代表关节的加速度，单位是 (°/S²)，用户可以自行填入适合的参数，加速度的值建议不要超过 720。
(e) joint_move 是阻塞的运动指令，必须一条运动指令执行完才会执行下一条运动指令。如果需要立即执行下一条指令，建议先使用 stop_program 停止当前的运动，再发送下一条运动指令。

4.8 TCP 末端运动到指定位置

发送消息： {"cmdName":"end_move","endPosition":[100,200.1,200.5,0,0,0], "speed":21.5, "accel":31.5}
接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "end_move"}

说明：

(a) endPosition: [x,y,z,a,b,c]　指定 TCP 末端 xyzabc 的值

(b) speed: speed_val 代表关节的速度，单位是 (°/S)，用户可以自行填入适合的参数。如果 speed_val 设置为 20,则代表关节速度为 20 °/S。

(f) accel：accel_val 代表关节的加速度，单位是 (°/S²)，用户可以自行填入适合的参数，加速度的值建议不要超过 720。

(c) end_move 是阻塞的运动指令，必须一条运动指令执行完才会执行下一条运动指令。如果需要立即执行下一条指令，建议先使用 stop_program 停止当前的运动，再发送下一条运动指令。

补充说明：

(a) end_move 指令并不是从当前位置直线运动到目标位置点，这条指令是先对用户输入的笛卡尔空间目标点进行逆解，然后使用 joint_move 指令，让机器人关节运动到指定位置。

(b) 如果需要从当前位置直线运动到目标位置点，需要使用 **moveL** 指令

4.9 直线运动 MoveL

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

接收消息：　{"errorCode": "0", "errorMsg": "", "cmdName": "moveL"}

说明：

(a) cartPosition: [x,y,z,rx,ry,rz]　指定笛卡尔空间 x,y,z,rx,ry,rz 的值

(b) speed: speed_val 代表线速度，单位是 (mm/s)，用户可以自行填入适合的参数。如果 speed_val 设置为 20,则代表线速度为 20 mm/s。

(c) accel:　accel_val 代表直线运动的加速度，单位是 (mm/S²)，用户可以自行填入适合的参数，加速度的值建议不要超过 8000.

(d) relFlag:　flag_val 可选值为 0 或者 1。 0 代表绝对运动，1 代表相对运动。

(e) moveL 是阻塞的运动指令，必须一条运动指令执行完才会执行下一条运动指令。如果需要立即执行下一条指令，建议先使用 stop_program 停止当前的运动，再发送下一条运动指令。

4.10 圆弧运动 MoveC

发送消息：

{"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}

接收消息：　{"errorCode": "0", "errorMsg": "", "cmdName": "movc"}

说明：

(a) pos_mid: [x,y,z,rx,ry,rz]　指定笛卡尔空间 x,y,z,rx,ry,rz 的值

(b) pos_end: [x,y,z,rx,ry,rz]　指定笛卡尔空间 x,y,z,rx,ry,rz 的值

(c) speed: speed_val 代表关节的速度，单位是 (°/S)，用户可以自行填入适合的参数。如果 speed_val 设置为 20,则代表关节速度为 20 °/S。

(d) accel：accel_val 代表关节的加速度，单位是 (°/S²)，用户可以自行填入适合的参数，加速度的值建议不要超过 720。

(e) tol：tol 代表最大允许误差，若不需要则取 0。

4.11 退出连接

发送消息： {"cmdName":"quit"}
接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "quit"}

4.12 获取机器人状态

机器人状态： robot_enabled/robot_disabled powered_on/powered_off
发送消息： {"cmdName":"get_robot_state"}
接收消息：
{"errorCode": "0", "errorMsg": "", "enable": "robot_disabled", "cmdName": "get_robot_state", "power": "powered_off"}
说明：
(a) enable：表示机器人的使能状态，robot_enabled/robot_disabled 分别是使能和未使能
(b) power：表示机器人的上电状态，powered_on/powered_off 分别是上电和未上电

4.13 伺服位置控制模式

relFlag: 1 代表进入 servo_move 模式，0 代表退出
发送消息： {"cmdName":"is_in_servomove"}
接收消息： {"errorCode": "0", "errorMsg": "", "in_servomove": true, "cmdName": "is_in_servomove"}
说明：
(a) in_servomove:true,处于伺服模式，false 不在伺服模式

4.14 机器人伺服位置控制使能

relFlag: 1 代表进入 servo_move 模式，0 代表退出
发送消息： {"cmdName":"servo_move","relFlag":0}
接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "servo_move"}

4.15 关节空间位置控制模式

relFlag: 0 代表绝对运动，1 代表相对运动
发送消息：{"cmdName":"servo_j","relFlag":1,"jointPosition":[0.1,0,0,0,0,0],"stepNum":1}
接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "servo_j"}
说明：
(a) 用户可以自己生成六个关节的角度然后发送给机器人，能控制机器人到达相应的目标位置
(b) 用户在使用 servo_j 指令之前需要先使用 servo_move 指令进入伺服位置控制模式
(c) jointPosition:[joint1,joint2,joint3,joint4,joint5,joint6] 填入的是各个关节的角度，单位是度。
(d) relFlag:0 relFlag 可选参数为 0 和 1 。0 代表绝对运动，1 代表相对运动。
(e) stepnum：是周期分频，机器人将以 num*8ms 的周期执行接收到的 servo 运动指令。
(f) 需要注意的是，由于控制器的控制周期是 8ms，这个指令需要 8ms 发送一次才能有效果，而且需要连续的发才行，只发一次看不出效果，并且六个关节的最大速度是 180 度/秒。
例如[1.5,0.5,0.5,0.5,0.5,0.5]，1.5/0.008= 187.5 超出了关节速度限制，那么 servo_j 指令就不会生效。

说明：这条指令和第 5 条指令 joint_move 的区别是比较大的，这条指令主要是科研当中做轨迹规划时使用，这

条指令发给机器人时并不经过控制器的规划器进行插补，而是直接发给伺服的。用户使用这条指令时需要预先进行轨迹的规划，否则使用效果会很差，无法达到预期。一般情况下建议使用 joint_move 指令。

4.16 笛卡尔空间位置控制模式

relFlag:　0 代表绝对运动，1 代表相对运动
发送消息：　{"cmdName":"servo_p","catPosition":[x,y,z,a,b,c],"relFlag":0,"stepNum":1}
接收消息：　{"errorCode": "0", "errorMsg": "", "cmdName": "servo_p"}
说明：
(a) 用户可以使用自己的轨迹规划算法，生成各个关节的点坐标，然后发送给机器人的六个关节，机器人能执行这些指令。
(b) 用户在使用 servo_j 指令之前需要先使用 servo_move 指令进入伺服位置控制模式
(c) catPosition:[x,y,z,a,b,c]　表示机器人在笛卡尔空间中的坐标 xyz 的单位是 mm，abc 的单位是度。
(d) abc 代表机器人姿态的欧拉角顺序是 XYZ
(e) relFlag:0　relFlag 可选参数为 0 和 1 。0 代表绝对运动，1 代表相对运动。
(g) stepnum：是周期分频，机器人将以 num*8ms 的周期执行接收到的 servo 运动指令。
(f) 需要注意的是，这个指令需要每 8ms 发送一次才能有效果。因为控制器是每 8ms 插补一个位置点。并且 x、y、z、a、b、c 的值不宜过大，避免超出控制器的安全速度限制。

补充说明：这条指令主要是科研当中做轨迹规划时使用，使用这条指令时，控制器的规划器不进行插补。用户使用这条指令时需要预先进行轨迹的规划，否则使用效果会很差，无法达到预期。一般情况下建议使用 moveL。

4.17 设置机器人速度倍率

发送消息：　{"cmdName":"rapid_rate","rate_value":0.2}
接收消息：　{"errorCode": "0", "errorMsg": "", "cmdName": "rapid_rate"}
说明：
(a) rate_value：　默认情况下机器人的程序运行倍率是 1，通过这个命令可以进行修改，设置范围是 0 到 1

4.18 加载程序

发送消息：{"cmdName":"load_program","programName":"track/test.jks"}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "load_program"}
说明：
(a) program_name 不支持中文。
(b) 如果控制器是 1.5 版本之前的，使用.ngc 作为文件结尾；1.5 及之后的版本以.jks 作为文件结尾；
(c) 若加载轨迹文件需要在文件名前加 track/

4.19 获取被加载程序名称

发送消息：{"cmdName":"get_loaded_program"}
接收消息：
{"errorCode": "0", "errorMsg": "", "cmdName": "get_loaded_program", "programName": "test.jks"}

4.20 运行程序

发送消息：{"cmdName":"play_program"}

接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "play_program"}

4.21 暂停程序

发送消息：{"cmdName":"pause_program"}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "pause_program"}

4.22 恢复程序

发送消息：{"cmdName":"resume_program"}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "resume_program"}

4.23 停止程序

发送消息：{"cmdName":"stop_program"}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "stop_program"}
说明：
(a) 停止程序实际上就是停止机器人的运动
(b) stop_program 也可以用于停止机器人的其它运动指令，比如 joint_move/moveL，当要控制机器人停止运动时建议使用 stop_program。

4.24 获取程序状态

程序状态：running/paused/idle
发送消息：{"cmdName":"get_program_state"}
接收消息：
{"errorCode": "0", "errorMsg": "", "cmdName": "get_program_state", "programState": "idle"}

4.25 设置数字输出变量(DO)的值

发送消息：{"cmdName":"set_digital_output","type":0,"index":1,"value":1}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_digital_output"}

说明：
(a) type 是 DO 的类型： 0 是控制器 DO，1 代表工具端 DO，2 代表扩展 DO
(b) index 是 DO 的编号，例如控制的 DO 编号为 1~31，如果想要控制第 31 个 DO，index 的值设为 31.
(c) value 是 DO 的值，可选值为 0 或者 1

4.26 获得数字输入状态

发送消息： {"cmdName":"get_digital_input_status"}
接收消息：
{"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], "errorCode": "0", "cmdName": "get_digital_input_status", "errorMsg": ""}
说明：

din_status:一共有 136 个引脚，前 8 个引脚为控制柜硬件 IO，第一个引脚被控制器占用，用户无法使用。之后的为 modbus

4.27 设模拟输出变量的值(AO)的值

发送消息：{"cmdName":"set_analog_output","type":type,"index":index,"value":value}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_analog_output"}
说明：
(a) type 是 AO 的类型 ：0 是控制器 AO，2 代表扩展 AO
(b) index 是 AO 的编号，例如控制的 AO 编号为 0~7，如果想要控制第 7 个 AO，index 的值设为 7.
(c) value 是 AO 的值，可输入一个浮点数来满足编程要求，例如 4.32

4.28 设置工具坐标系

发送消息：{"cmdName":"set_tool_offsets","tooloffset":[10,10,10,10,10,10],"id":2,"name":"tcp_new"}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_tool_offsets"}
说明：
(a) tooloffset: 填入工具坐标系的 x, y, z, rx, ry, rz。如例子中的[10,10,10,10,10,10]
(b) id: 想要设置的工具坐标系的 ID 号，可选的 ID 为 1 到 10。例入设置工具坐标系 2。
(c) name: 想要设置的工具坐标系的名字。例如将名字设置为 tcp_new。

4.29 选择工具坐标系

发送消息：{"cmdName":"set_tool_id","tool_id":2}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_tool_id"}
说明：
(a) tool_id: 工具坐标系 ID 号。用来选择在哪个工具坐标系下运动。比如设置 ID 为 2，就是在第二个工具坐标系下运动。
(b) 需要说明的是当 tool_id 为 0 的时候机器人的工具默认为末端法兰盘中心。

4.30 获取工具坐标系信息

发送消息：{"cmdName":"get_tool_offsets"}
接收消息：{"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"}
说明：
(c) tool_offset:有 11 组数据（0~10），对应相对 id 的工具坐标系偏移。

4.31 设置用户坐标系

发送消息：{"cmdName":"set_user_offsets","userffset":[10,10,10,10,10,10],"id":2,"name":"user_new"}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_user_offsets"}
说明：
(a) userffset: 填入用户坐标系的 x, y, z, rx, ry, rz 如例子中的[10,10,10,10,10,10]。
(b) id: 想要设置的用户坐标系的 ID 号，可选的 ID 为 1 到 10。例入设置用户坐标系 2。

(c) name: 想要设置的用户坐标系的名字。例如将名字设置为 user_new。

4.32 选择用户坐标系

发送消息：{"cmdName":"set_user_id","user_frame_id":2}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_user_id"}
说明：
(a) set_user_id: 用户坐标系 ID 号。用来选择在哪个用户坐标系下运动。比如设置 ID 为 2，就是在第二个用户坐标系下运动。
(b) 需要说明的是 set_user_id 设置为 0 的时候代表机器人在基坐标系下运动。

4.33 获取用户坐标系信息

发送消息：{"cmdName":"get_user_offsets"}
接收消息：{"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": ""}
说明：
(a) tool_offset:有 11 组数据（0~10），对应相对 id 的工具坐标系偏移。

4.34 获得扩展 IO 状态

发送消息： {"cmdName":"get_extio_status"}
接收消息：
无扩展模块时:
(0, [], {'state': 0})
有扩展模块时:
{"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)})"}
说明：
对 extio_status 进行说明
(a) 红色字体的数字 1 代表，有 1 个外部扩展 IO 模块。
(b) setup: (1, 'mobusIoName', ('192.168.2.47', 8888) 1 代表 TCP(0 代表 RTU)，modbusIoName 是外部扩展模块的名字，('192.168.2.47', 8888) 是 IP 和端口。
(c) {'ai': (0, 0), 'do': (0, 0), 'ao': (0, 0), 'di': (0, 8)} : (0,0) 括号内第一个数字代表外部扩展模块寄存器起始分配地址。第二个数字代表 IO 的数量。
(d) pinmap: 外部模块引脚索引起始地址,当有多个外部模块的时候需要考虑。
(e) {'state': 0, 'di': (0, 0, 0, 0, 0, 0, 0, 0)})"}: state 为 0 代表扩展模块不在运行，1 代表扩展模块正在运行。

4.35 获得功能输入引脚的状态

发送消息：{"cmdName":"get_funcdi_status"}
接收消息：{"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"}

说明：

(a) funci_status: 一共返回 12 中功能输入引脚，分别是

(1) 启动程序 (2) 暂停程序 (3) 继续运行程序

(4) 停止程序 (5) 打开电源 (6) 关闭电源

(7) 机器人上使能 (8) 机器人下使能 (9) 一级缩减模式

(10)保护性停止 (11)回初始位置 (12)二级缩减模式

(b) [type,index] :[-1,-1]为默认值表示没有输入引脚被设置为功能引脚。Type 的值有三种，0 代表控制柜输入引脚，1 代表机器人末端工具输入引脚，2 代表 modbus 的输入引脚。Index 代表的是输入引脚的索引号，表示哪一个引脚被设置为功能引脚。如上文中[0,2]表示控制柜的第三个引脚(索引号从 0 开始)被设置为一级缩减模式功能引脚。

4.36 设置拖拽拖拽模式开关

发送消息：{"cmdName":"torque_control_enable","enable_flag":0}

接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "torque_control_enable"}

说明：

(a) enable_flag，0 关闭拖拽模式，1 开启拖拽模式

4.37 获得拖拽状态

发送消息：{"cmdName":"drag_status"}

接收消息：{"drag_status": "False", "errorCode": "0", "cmdName": "drag_status", "errorMsg": ""}

说明：

(a) drag_status 为 True 代表机器人正处于拖拽模式下，用户可以对机器人进行拖拽编程。值为 False 则相反。

4.38 查询用户自定义系统变量

发送消息：{"cmdName":"query_user_defined_variable"}

接收消息：{"errorCode": "0", "errorMsg": "", "var_list": [{"alias": "test", "id": 5500, "value": 1.0}], "cmdName": "query_user_defined_variable"}

说明：

(a) var_list: 会以列表的形式返回所有的用户自定义的系统变量

(b) alias: 是系统变量的名字 id:是系统变量的 id 号，用户想要修改系统变量需要知道系统变量的 id 号。value:是系统变量的值。例如返回消息中返回的系统变量名字是 system_var1，id 是 5500，值为 12.0

4.39 修改用户自定义系统变量

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

接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "modify_user_defined_variable"}

说明：

(a) id_new:是要修改的系统变量的 id 号，id 号可以通过指令 query_user_defined_variable 查询获得

alias_new:设置要修改的系统变量的新名字 value_new:设置要修改的系统变量的值

4.40 保护性停止状态

发送消息：{"cmdName":"protective_stop_status"}

接收消息：{"errorCode": "0", "errorMsg": "", "protective_stop": "0", "cmdName": "protective_stop_status"}

(a) protective_stop 的值为 1 时代表机器人处于保护性停止状态，值为 0 的时候相反。保护性停止一般发生在机器人发生碰撞的情况下。这时候机器人会处于保护性停止状态。

4.41 wait_complete 指令

发送消息：{"cmdName":"wait_complete"}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": " wait_complete"}
说明：**本条指令已经废弃，不建议使用。**

4.42 设置负载

发送消息：{"cmdName":"set_payload","mass":m,"centroid":[x,y,z]}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": " set_payload "}

说明：mass 是负载的质量，单位是千克(KG)。centroid 是负载的质心，单位是 mm。
例如要设置的负载质量为 1KG，质心为[10,10,10]。那么可以发送命令:
{"cmdName":"set_payload","mass":1,"centroid":[10,10,10]}

4.43 获取负载

发送消息：{"cmdName":"get_payload"}
接收消息：{"errorCode": "0", "centroid": [0.0, 0.0, 1.0], "mass": 1.0, "cmdName": "get_payload", "errorMsg": ""}
说明：mass 是负载的质量，单位是千克(KG)。 centroid 是负载的质心，单位是 mm。

4.44 设置机器人碰撞等级

发送消息：{"cmdName":"set_clsn_sensitivity","sensitivityVal":level}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_clsn_sensitivity"}

说明：level 的可选值为 0 到 5。
0 代表关闭碰撞。
等级 1 的碰撞灵敏度是最高的, 25N。
等级 5 的碰撞灵敏度是最低的, 125N,每级差 25N。

4.45 获取机器人碰撞等级

发送消息：{"cmdName":"get_clsn_sensitivity"}
接收消息：{"errorCode": "0", "sensitivityLevel": level, "cmdName": "get_clsn_sensitivity", "errorMsg": ""}

说明：level 是返回的碰撞等级
0 代表关闭碰撞。
1 代表的碰撞灵敏度是最高的。
5 代表的碰撞灵敏度是最低的。

4.46 机器人运动学正解

发送消息：{"cmdName":"kine_forward","jointPosition":[j1,j2,j3,j4,j5,j6]}

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

说明：[j1,j2,j3,j4,j5,j6]是需要发送的 6 个关节角度用来求运动学正解，
[x,y,z,rx,ry,rz] 是经过正解后得到的机器人末端位姿。

4.47 机器人运动学逆解

发送消息：
{"cmdName":"kine_inverse","jointPosition":[j1,j2,j3,j4,j5,j6],"cartPosition": [x,y,z,rx,ry,rz]}
接收消息：
{"errorCode": "0", "errorMsg": "", "cmdName": "kine_inverse", "jointPosition":[x,y,z,rx,ry,rz]}
若逆解失败·:{"errorCode": "2", "errorMsg": "call kine_inverse failed", "cmdName": "kine_inverse"}

说明：

(a) 发送的 jointPosition 是机器人的参考关节角，推荐用户选取机器人当前的关节角作为参考关节角。单位是度。
(b) 发送消息里的 cartPosition 是机器人的末端位姿。[x,y,z,a,b,c] xyz 的单位是毫米，abc 的单位是度。
(c) 接收消息里得到的 jointPosition，是逆解后的机器人关节角度。

4.48 碰撞后从碰撞保护模式恢复

发送消息：{"cmdName":"clear_error"}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "clear_error"}

4.49 获取关节位置

发送消息：{"cmdName":"get_joint_pos"}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "get_joint_pos", "joint_pos": [j1,j2,j3,j4,j5,j6]}

说明：j1,j2,j3,j4,j5,j6 是返回的六个关节的位置，单位是度

4.50 获取当前坐标系下的末端位姿

发送消息：{"cmdName":"get_tcp_pos"}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": " get_tcp_pos", "tcp_pos": [x,y,z,a,b,c]}

说明：[x,y,z,a,b,c] 是在当前设置的工具坐标系下的末端位姿，默认情况下工具坐标系是法兰盘中心。

4.51 获取控制器版本

发送消息：{"cmdName":"get_version"}
接收消息：{"errorCode": "0", "errorMsg": "", "version": "1.6.5_01_X86", "cmdName": "get_version"}

4.52 设置轨迹采集参数

发送消息：{"cmdName":"set_traj_config","acc":100,"vel":20,"xyz_interval":0.1,"rpy_interval":0.1}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_traj_config"}

说明：

(a) acc: 轨迹执行的加速度 mm/s^2。

(b) vel: 轨迹执行的速度 mm/s.

(c) xyz_interval: 笛卡尔空间位移数据采集间隔。

(d) rpy_interval: 笛卡尔空间旋转数据采集间隔。

4.53 获取轨迹采集参数

发送消息：{"cmdName":"get_traj_config"}

接收消息：{"acc": 100.0, "xyz_interval": 0.1, "errorMsg": "", "cmdName": "get_traj_config", "errorCode": "0", "rpy_interval": 0.1, "vel": 20.0}

说明：

(a) acc: 轨迹执行的加速度 mm/s^2。

(b) vel: 轨迹执行的速度 mm/s.

(c) xyz_interval: 笛卡尔空间位移数据采集间隔。

(d) rpy_interval: 笛卡尔空间旋转数据采集间隔。

4.54 设置轨迹采集开关

发送消息：{"cmdName":"set_traj_sample_mode","mode":1,"filename":"test"}

接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_traj_sample_mode"}

说明：

(a) mode: 0 关闭，1 开启。

(b) filename: 文件名.

4.55 获取轨迹采集开关状态

发送消息：{"cmdName":"get_traj_sample_status"}

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

说明：

(a) sampleStatus: 0 关闭，1 开启。

4.56 查询控制器中已经存在的轨迹复现数据的文件名

发送消息：{"cmdName":"get_exist_traj_file_name"}

接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "get_exist_traj_file_name", "trajTrackFileName": ["pythonTrack1", "test"]}

说明：trajTrackFileName：轨迹数据文件名称

4.57 重命名轨迹复现数据的文件名

发送消息：{"cmdName":"rename_traj_file_name","src":"test","dest":"test_rename"}

接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "rename_traj_file_name"}

说明：

(a) src: 源轨迹数据文件名

(b) dest: 修改成的文件名.

4.58 删除轨迹

发送消息：{"cmdName":"remove_traj_file","fileName":fileName}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "remove_traj_file"}
说明：
(a) filename: 轨迹数据文件名

4.59 控制器中轨迹复现数据文件生成控制器执行脚本

发送消息：{"cmdName":"generate_traj_exe_file","fileName":"test_rename"}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "generate_traj_exe_file"}
说明：
(a) filename: 轨迹数据文件名

4.60 伺服模式滤波器

set_servo_move_filter 指令设置了伺服模式滤波器，协助规划轨迹
filter_type 代表运动模式可填的值有 6 个:

no_filter 禁用滤波器:	0
关节空间一阶低通滤波器:	1
关节空间非线性滤波器:	2
关节空间多阶均值滤波器:	3
笛卡尔空间非线性滤波器:	4
速度前瞻:	5

当 filter_type **为 0** 的时候只有 1 个参数
发送消息： {"cmdName":"set_servo_move_filter","filter_type":0}
接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "set_servo_move_filter"}
说明：
禁用滤波器，不使用滤波器进行辅助规划，需要退出 servo 模式后设置。
当 filter_type **为 1** 的时候只有 2 个参数
发送消息： {"cmdName":"set_servo_move_filter","filter_type":1",lpf_cf":0.5}
接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "set_servo_move_filter"}
说明：
(a) 设置关节空间一阶低通滤波器，需要退出 servo 模式后设置
(b) lpf_cf 表示截止频率，单位 HZ。
当 filter_type **为 2** 的时候只有 4 个参数
发送消息： {"cmdName":"set_servo_move_filter","filter_type":2,"nlf_max_vr":2,"nlf_max_ar":2,"nlf_max_jr":4}
接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "set_servo_move_filter"}
说明：
(a) 设置关节空间非线性滤波器，需要退出 servo 模式后设置
(b) max_vr 姿态变化速度的速度上限值（绝对值）°/s
(c) max_ar 姿态变化速度的加速度上限值（绝对值）°/s^2
(d) max_jr 姿态变化速度的加加速度上限值（绝对值）°/s^3
当 filter_type **为 3** 的时候只有 5 个参数
发送消息：

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

接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "set_servo_move_filter"}

说明：

(a) 设置关节空间多阶均值滤波器，需要退出 servo 模式后设置

(b) mmf_max_buf: 滤波窗口宽度，越大滤波轨迹越平缓，会损失精度

(c) mmf_kp：位置跟踪系数，越小滤波轨迹越平缓，越大精度越高，但可能有抖动

(d) mmf_kv:速度跟踪系数越小滤波轨迹越平缓，越大精度越高，但可能有抖动

(e) mmf_ka： 加速度位置跟踪系数越小滤波轨迹越平缓，越大精度越高，但可能有抖动

当 filter_type **为 4** 的时候只有 7 个参数

发送消息：

{"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}

接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "set_servo_move_filter"}

说明：

(a) 设置笛卡尔空间非线性滤波器，需要退出 servo 模式后设置

(b) max_vr 笛卡尔空间姿态变化速度的速度上限值（绝对值）°/s

(c) max_ar 笛卡尔空间姿态变化速度的加速度上限值（绝对值）°/s^2

(d) max_jr 笛卡尔空间姿态变化速度的加加速度上限值（绝对值）°/s^3

(e) max_vp 笛卡尔空间下移动指令速度的上限值（绝对值）。单位：mm/s

(f) max_ap 笛卡尔空间下移动指令加速度的上限值（绝对值）。单位：mm/s^2

(g) max_jp 笛卡尔空间下移动指令加加速度的上限值（绝对值）单位：mm/s^3

当 filter_type **为 5** 的时候只有 3 个参数

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

接收消息： {"errorCode": "0", "errorMsg": "", "cmdName": "set_servo_move_filter"}

说明：

(a) 设置关节空间多阶均值滤波器，需要退出 servo 模式后设置

(b) max_buf 均值滤波器缓冲区的大小

(c) mmf_kp 加速度滤波系数

4.61 设置传感器品牌

发送消息：{"cmdName":"set_torsenosr_brand","sensor_brand":num}

接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_torsenosr_brand"}

说明：

(a) sensor_band:1 为 SONY 索尼半导体；2 为 BoschSensortec 博世；3 为 ST 意法半导体

4.62 获取传感器品牌

发送消息：{"cmdName":"get_torsenosr_brand"}

接收消息：{"sensor_brand": 1, "errorCode": "0", "cmdName": "get_torsenosr_brand", "errorMsg": ""}

说明：

(a) sensor_band:1 为 SONY 索尼半导体；2 为 BoschSensortec 博世；3 为 ST 意法半导体

4.63 设置柔顺控制参数

发送消息：
{"cmdName":"set_admit_ctrl_config","axis":2,"opt":1,"ftUser":25,"ftConstant":5,"ftNnormalTrack":0,"ftReboundFK":0}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_admit_ctrl_config"}
说明：
(a) axis:0:x 轴；1：y 轴；2：z 轴；3：rx4：ry5：rz
(b) opt：0 关 1 开
(c) ftUser：阻尼力，表示用户用多大的力才能让机器人的沿着某个方向以最大速度进行运动
(d) ftReboundFK 回弹力，表示机器人回到初始状态的能力
(e) ftConstant 代表恒力，手动操作时全部设置为 0
(f) ftNnormalTrack 法向跟踪，手动操作时全部设置为 0

4.64 获取力控柔顺控制参数

发送消息：{"cmdName":"get_admit_ctrl_config"}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "get_admit_ctrl_config", "admitConfig": [[0, 5.0, 0.0, 0.0, 0], [0, 5.0, 0.0, 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]]}
说明：
(a) admitConfig : 有 6 组 1*5 的矩阵分别代表 x,y,z,rx,ry,rz 的力控柔顺控制参数.
(b) 每个矩阵内按顺序[opt,ftUser,ftReboundFK ,ftConstant ,ftNnormalTrack].

4.65 设置力矩传感器开关

发送消息：{"cmdName":"set_torque_sensor_mode","sensor_mode":mode}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_torque_sensor_mode"}
说明：
(a) mode:0 开 1 关

4.66 开始辨识工具末端负载

发送消息：{"cmdName":"start_torq_sensor_payload_identify","jointPosition":[j1,j2,j3,j4,j5,j6]}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "start_torq_sensor_payload_identify"}
说明：
(a) 这是一条运动指令，运动到 jointposition 指定的位置。
(b) jointPosition：负载辨识终点，推荐 456 轴各旋转 90 度,以得到准确的辨识结果

4.67 获取末端负载辨识状态

发送消息：{"cmdName":"get_torq_sensor_identify_staus"}
接收消息:{"errorCode": "0", "errorMsg": "", "cmdName": "get_torq_sensor_identify_staus", "identify_status": 1}
说明：
(a) identify_status:0 代表辨识完成，1 代表未完成，2 代表辨识失败

4.68 获取末端负载辨识结果

发送消息：{"cmdName":"get_torq_sensor_payload_identify_result"}
接收消息：{"errorCode": "0", "centroid": [0, 0, 0], "mass": 0, "cmdName": "get_torq_sensor_payload_identify_result", "errorMsg": ""}
说明：
(a) mass 是负载的质量，单位是千克(KG)。
(b) centroid 是负载的质心，单位是 mm。

4.69 设置传感器末端负载

发送消息：{"cmdName":"set_tool_payload","mass":weight,"centroid":[x,y,z]}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_tool_payload"}
说明：mass 是负载的质量，单位是千克(KG)。centroid 是负载的质心，单位是 mm。
例如要设置的负载质量为 1KG，质心为[10,10,10]。那么可以发送命令:
{"cmdName":"set_payload","mass":1,"centroid":[10,10,10]}

4.70 获取传感器末端负载

发送消息：{"cmdName":"get_tool_payload"}
接收消息：{"errorCode": "0", "centroid": [0.0, 0.0, 0.0], "mass": 1.0, "cmdName": "get_tool_payload", "errorMsg": ""}
说明：
(a) mass 是负载的质量，单位是千克(KG)。
(b) centroid 是负载的质心，单位是 mm。

4.71 设置柔顺控制力矩条件

发送消息：{"cmdName":"set_compliance_condition","compliant_condition":[0,0,0,0,0,0]}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_compliance_condition"}
说明：
(a) compliant_condition:力传感器的受力分量和力矩分量，fx:沿 x 轴受力分量;tx:绕 x 轴力矩分量

4.72 设置导纳控制开关

发送消息：{"cmdName":"enable_admittance_ctrl","enable_flag":1}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "enable_admittance_ctrl"}
说明：
(a) enable_flag: 0 关 1 开。

4.73 获取力控类型和传感器初始化状态

发送消息：{"cmdName":"get_compliant_type"}
接收消息：{"errorCode": "0", "errorMsg": "", "compliance_type": 0, "sensor_compensation": 1, "cmdName": "get_compliant_type"}
说明：
(a) sensor_compensation 是否开启传感器补偿,1 代表开启即初始化,0 代表不初始化

(b) compliance_type 0 代表不使用任何一种柔顺控制方法 1 代表恒力柔顺控制,2 代表速度柔顺控制

4.74 设置力控类型和传感器初始化状态

发送消息：{"cmdName":"set_compliant_type","sensor_compensation":1,"compliance_type":1}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_compliant_type"}
说明：
(a) sensor_compensation: 开启传感器补偿标志，1 代表开启初始化，0 代表不初始化
(b) compliance_type: 力控类型，0 代表不使用任何一种柔顺控制方法 1 代表恒力柔顺控制,2 代表速度柔顺控制

4.75 设置网络异常运动情况

发送消息：{"cmdName":"set_network_exception_handle","timeLimit":100,"motType":0}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_network_exception_handle"}
说明：
(a) millisecond 时间参数，单位毫秒,网络异常后延时多少 ms 触发
(b) mnt 网络异常时机器人需要进行的动作类型:0 保持；1 暂停；2 终止

4.76 设置力矩传感器 IP 地址

发送消息：{"cmdName":"set_torque_sensor_comm","type":0,"ip_addr":"172.30.1.110","port":55555}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_torque_sensor_comm"}
说明：
(a) type: 0 为使用 tcp/ip 协议，1 为使用 RS485 协议
(b) ip_addr 为力控传感器地址
(c) port 为使用 tcp/ip 协议时力控传感器端口号

4.77 获取力矩传感器 IP 地址

发送消息：{"cmdName":"get_torque_sensor_comm"}
接收消息：{"ip_addr": "192.168.2.228", "errorMsg": "", "cmdName": "get_torque_sensor_comm", "errorCode": "0", "type": 0, "port": 49152}
说明：
(a) type: 0 为使用 tcp/ip 协议，1 为使用 RS485 协议
(b) ip_addr: 力控传感器地址
(c) port: 使用 tcp/ip 协议时力控传感器端口号

4.78 关闭力矩控制

发送消息：{"cmdName":"disable_force_control"}
接受消息: {"errorCode": "0", "errorMsg": "", "cmdName": "disable_force_control"}

4.79 设置速度柔顺控制参数

发送消息：{"cmdName":"set_vel_compliant_ctrl","compliant_ctrl":[vc_level, rate1, rate2, rate3, rate4]}
接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_vel_compliant_ctrl"}

说明：

(a) compliant_ctrl:

vc_level 速度柔顺控制等级

rate1 比率等级 1

rate2 比率等级 2

rate3 比率等级 3

rate4 比率等级 4

4.80 设置 tioV3 工具电压

发送消息：{"cmdName":"set_tio_vout_param","tio_vout_ena":0,"tio_vout_vol":0}

接收消息：{"errorCode": "0", "errorMsg": "", "cmdName": "set_tio_vout_param"}

说明：

(a) tio_vout_ena 参数为 0 或 1 ；0 为已使能，1 为未使能

(b) tio_vout_vol 参数为 0 或 1；0 为 12V，1 为 24V

4.81 获取 tioV3 工具电压

发送消息：{"cmdName":"get_tio_vout_param"}

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

(a) tio_vout_ena 参数为 0 或 1；0 为已使能，1 为未使能

(b) tio_vout_vol 参数为 0 或 1；0 为 12V，1 为 24V

4.82 获取机器人 DH 参数

发送消息：{"cmdName":"get_dh_param"}

接收消息：{"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"}

第 5 章 错误描述

errorCode	errorMsg	Description
0	返回消息为空	消息执行成功
1	Exception:function call failed	程序发生异常
2	返回具体的错误消息	错误信息

第 6 章 JSON 介绍

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。简单地说，JSON 可以将 JavaScript 对象中表示的一组数据转换为字符串，然后就可以在函数之间轻松地传递这个字符串，或者在异步应用程序中将字符串从 Web 客户机传递给服务器端程序。这个字符串看起来有点儿古怪，但是 JavaScript 很容易解释它，而且 JSON 可以表示比“名称/值对”更复杂的结构。

JSON 语法是 JavaScript 对象表示语法的子集。

- 数据在名称/值对中；
- 数据由逗号分隔；
- 花括号保存对象；
- 方括号保存数组；

JSON 值可以是：数字（整数或浮点数）、字符串（在双引号中）、逻辑值（true 或 false）、数组（在方括号中）、对象（在花括号中）、null。

第 7 章 参考代码

请参考附件的 python 代码。代码中对关键部分做了注释。Python 文件为 TCP 命令使用的 demo。使用时需要把文件中的 IP 地址改为自己的机器人的 IP 地址，端口号是固定的不需要修改。

第 8 章　反馈和勘误

文档中如若出现不准确的描述或者错误，恳请读者指正批评。如果您在阅读过程中发现任何问题或者有想提出的意见，可以发送邮件到 support@jaka.com 我们的同事会尽量一一回复。

节卡机器人股份有限公司
地址：上海市闵行区剑川路 610 号 33-35 幢
电话：400-006-2665
网址：www.jaka.com