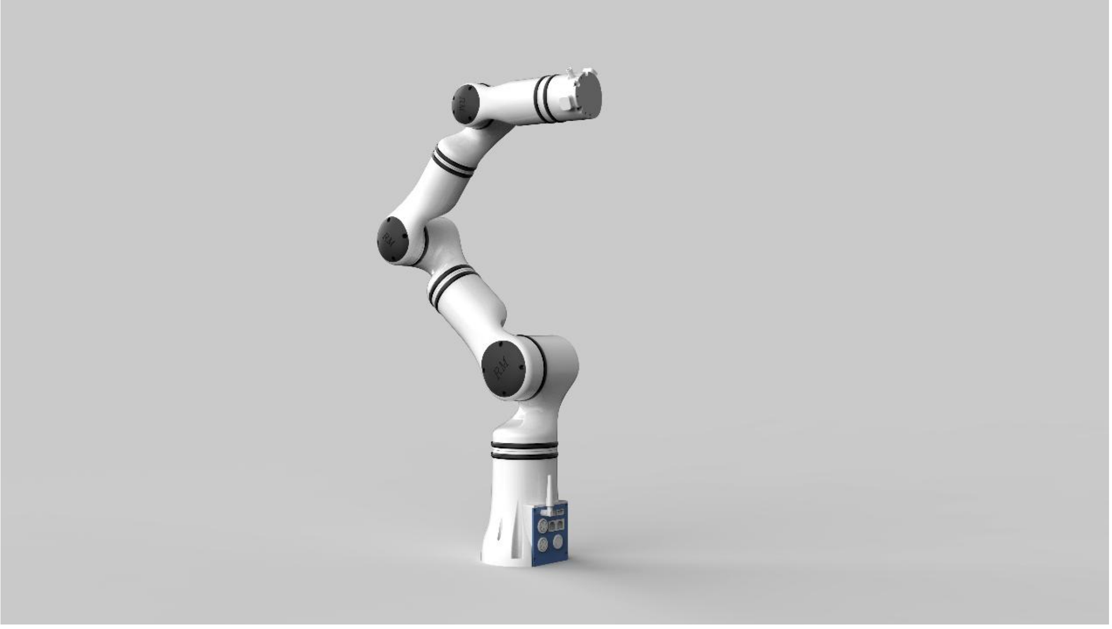
RM 6-DoF Robot JSON Protocol (V3.1)



RM Intelligent Technology (Beijing) Co., Ltd.

Version History

Version	Date	Comment
V1.0	2020-05-01	Init
V1.1	2020-05-10	Fix known errors
V1.2	2020-05-15	Fix known errors (Generalized revision)
V1.3	2020-05-17	Simplify some return values
V1.4	2020-05-20	Modify some punctuation marks
V1.4.1	2020-05-25	Modify some format
V1.4.2	2020-06-5	Modify the WIFI configuration procedure
V1.4.3	2020-06-18	Modify the return frames of *arm_all_state*
V1.4.4	2020-06-29	Modify the communication error message
V1.4.5	2020-06-29	Add the IO protocol
V1.4.6	2020-07-03	Modify some command names
V1.4.7	2020-07-28	Add the section of drag teaching
V1.4.8	2020-08-02	Add the interface of the arm end
V1.5	2021-03-12	Added the center of mass of the arm end, path point cache and other features
V1.6	2021-05-20	Add Movej_P command, PWM setting and one-axis force setting
V1.7	2021-07-26	Add the dynamic calibration parameter download command
V1.8	2021-08-18	Add the Modbus protocol configuration for the controller and end interface board
V1.9	2021-09-26	Software versions that can display multiple cores
V2.0	2021-09-27	Add collision avoidance level setting
V2.1	2021-11-20	Add joint calibration setting
V2.2	2022-01-12	Add online programming setting
V2.3	2022-01-28	Add one-axis force setting
V2.4	2022-02-12	Explicate RS485 usage
V2.5	2022-02-15	Add one click/button to set joint limits
V2.6	2022-02-19	Fix known errors
V2.7	2022-03-07	Add passthrough hybrid force-position compensation; Modify angle and pose passthrough
V2.8	2022-04-08	Add high-speed network port control
V2.9	2022-04-14	Modify the lift return status
V3.0	2022-05-06	Mask test instruction
V3.1	2022-05-13	Optimize document format

Contents

RealMan (RM) 6-DoF robot arm or robot adopts the unified JSON format for data communication. Users can use WIFI (AP or STA) and Ethernet port to send data in JSON format through the standard TCP/IP communication protocol to control the robot arm.

Alternatively, the user can control the robot arm by sending a JSON format string through the RS485 interface or RS232-USB interface of the robot arm controller. Both interfaces have a default baud rate of 460800BPS, stop bit of 1, data bit of 8, and no check bits.

All the above control modes do not need to be manually switched by the user, the robot arm will automatically recognize them. The robot arm will return the command in the same way after the user sends the command in the specified format through either mode.

Note: All data must be sent as a new line, that is, the instruction ends with "\r\n". Otherwise, the robot arm will not respond.

Note: The above communication modes cannot be used simultaneously to avoid mutual interference of commands. At the same time, when users use JSON protocol for development and testing, please do not connect to the teach pendant to prevent command conflict.

Note: RM robot arm is equipped with high-speed network port, users can open the high-speed network port through the command for the development of tasks with high real-time requirements.

1. Joint Configuration

1.1 The Command Set for the Joint Configuration

If the joint returns error, the joint hyperparameters cannot be modified. The joint error code must be cleared first. The joint must be disenabled before set the joint, otherwise it's not going to work.

Note: All hyperparameters of the joint will be automatically saved to the Flash memory of the joint after modification, which will take effect immediately. After that, the joint will be in the disenabled status. After modification, commands must be sent to enable the joint.

Note: All parameters of the RM robot arm have been configured to the best state before leaving the factory, and it is not recommended that the user modify the underlying parameters of the joint. If the user really needs to modify, first of all, the robot arm should be in a disenabled state, and then send a modified parameter instruction, after the parameter setting is successful, send the joint recovery enable instruction. It should be noted that when the joint is restored, the user needs to ensure that the joint is in a static state to avoid positioning errors in the joint during the activation process. After the joint is enable, the user can control the joint movement.

(1) Set the maximum rotation speed of the joint

Functionality	To set the maximum rotation speed of the joint.
Argument description	set_joint_max_speed: Set the maximum rotation speed of the joint. joint_max_speed: Joint ID and its max rotation speed. Unit: RPM.
Command format	{s:s, s:[i, i]}
Demo	{"command":"set_joint_max_speed","joint_max_speed":[2, 300000]}
	Explain: Set Joint2, max rotation speed of 300RPM, rotation speed resolution of 0.001RPM.
Return	Format: {s:s, s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_joint_max_speed","joint_max_speed":true}

(2) Set the maximum acceleration of the joint

Functionality	To set the maximum acceleration of the joint.
Argument description	set_joint_max_acc: Set the maximum acceleration of the joint. joint_max_acc: Joint ID and its max acceleration. Unit: RPM/s.
Command format	{s:s, s:[i, i]}
Demo	{"command":"set_joint_max_acc","joint_max_acc":[2, 30000]}
	Explain: Set Joint2, max acceleration of 30RPM/s, acceleration resolution of 0.001RPM/s.
Return	Format: {s:s, s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_joint_max_acc","joint_max_acc": true }

(3) Set the minimum angle of the joint

Functionality	To set the minimum angle of the joint.
Argument description	set_joint_min_pos: Set the minimum angle of the joint. joint_min_pos: Joint ID and its min angle. Unit: degree(°).
Command format	{s:s, s:[i, i]}
Demo	{"command":"set_joint_min_pos","joint_min_pos":[1, -170000]} Explain: Set Joint1, min angle of -170°, angle resolution of 0.001°.
Return	Format: {s:s, s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_joint_min_pos","joint_min_pos": true }

(4) Set the maximum angle of the joint

Functionality	To set the maximum angle of the joint.
Argument description	set_joint_max_pos: Set the maximum angle of the joint. joint_max_pos: Joint ID and its max angle. Unit: degree(°).
Command format	{s:s, s:[i,i]}
Demo	{"command":"set_joint_max_pos","joint_max_pos":[1, 170000]} Explain: Set Joint1, max angle of 170°, angle resolution of 0.001°.
Return	Format: {s:s, s:b}, true- Setup succeeded, false- Setup failed.
	{"command":"set_joint_ max _pos","joint_ max _pos":true}

(5) Set the enable status of the joint

Functionality	To set the status of being enabled or not of the joint.
Argument description	set_joint_en_state: Set the enable status of the joint. joint_en_state: Joint ID and its enable status,1: enable, 0: disenable.

Command format	{s:s, s:[i, i]}
Demo	{"command":"set_joint_en_state","joint_en_state":[6, 1]} Explain: Enable Joint6.
Return	Format: {s:s, s:b}, true- Setup succeeded, false- Setup failed.
	{"command":"set_joint_en_state","joint_en_state":true}

(6) Set the origin of the joint

Functionality	To set the origin of the joint.
Argument description	set_joint_zero_pos: Set the origin of the joint. joint_zero_pos: Joint ID.
Command format	{s:s, s:i}
Demo	{"command":"set_joint_zero_pos","joint_zero_pos":3} Explain: Set the position of Joint3 as its origin.
Return	Format: {s:s, s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_joint_zero_pos","joint_zero_pos":true}

(7) Clear joint error code

Functionality	To clear the error code of the joint.
Argument description	set_joint_clear_err: Clear the error code of the joint. joint_clear_err: Joint ID.
Command format	{s:s, s:i}
Demo	{"command":"set_joint_clear_err","joint_clear_err":2} Explain: clear the error code of Joint2.
Return	Format: {s:s, s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_joint_clear_err","joint_clear_err":true}

(8) One click/button to set joint limits

Functionality	This function is only for administrators, not for normal users. Its purpose is to switch the joint limits pre-assembling mode and product mode.
Argument description	auto_set_joint_limit
Command format	{s:s,s:i}
Demo	{"command":"auto_set_joint_limit","limit_mode":1} Explain: limit_mode：limit mode. 0- Pre-assembling mode，each joint limit is ±720°. 1- Product mode，each joint limit is the limit in the specification.
Return	Format：{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"auto_set_joint_limit","set_state ":true}

1.2 The Query Set of Joint Configuration

(1) Query joint maximum speed

Functionality	To query the maximum speed of the joint.
Argument description	get_joint_max_speed: query the max speed of the joint.
Command format	{s:s}

Demo	{"command":"get_joint_max_speed"}
	Explain: query the max speed of the joint.
Return	See Table 1.3-(1)

(2) Query joint maximum acceleration

Functionality	To query the maximum acceleration of the joint.
Argument description	get_joint_max_acc: query the maximum acceleration of the joint.
Command format	{s:s}
Demo	{"command":"get_joint_max_acc"}
	Explain: query the maximum acceleration of the joint.
Return	See Table 1.3-(2)

(3) Query joint minimum angle

Functionality	To query the minimum angle of the joint.
Argument description	get_joint_min_pos: query the minimum angle of the joint.
Command format	{s:s}
Demo	{"command":"get_joint_min_pos"}
	Explain: query the minimum angle of the joint.
Return	See Table 1.3-(3)

(4) Query joint maximum angle

Functionality	To query the maximum angle of the joint.
Argument description	get_joint_max_pos: query the maximum angle of the joint.
Command format	{s:s}
Demo	{"command":"get_joint_max_pos"}
	Explain: query the maximum angle of the joint.
Return	See Table 1.3-(4)

(5) Query the enable status of the joint

Functionality	To query the status of being enabled or not of the joint.
Argument description	get_joint_en_state: query the enable status of the joint.
Command format	{s:s}
Demo	{"command":"get_joint_en_state"}
	Explain: query the enable status of the joint.
Return	See Table 1.3-(5)

(6) Query joint error code

Functionality	To query the error code of the joint.
Argument description	get_joint_err_flag: Query the error code of the joint.
Command format	{s:s}
Demo	{"command":"get_joint_err_flag"}
	Explain: query the error code of the joint.
Return	See Table 1.3-(6)

1.3 The Return Set of Joint Configuration

(1) Return joint maximum speed

Functionality	To return the maximum speed of all joints.
Argument description	joint_max_speed: Return the max speed of the joints.
Command format	{s:s,s:[i,i,i,i,i,i]}
Demo	{"state":"joint_max_speed", "joint_speed":[30, 30, 30, 30, 30, 30]}
	Explain: Return the max rotation speed of the 6 joints, speed: 0.03RPM, unit: RPM, resolution: 0.001RPM.

(2) Return joint maximum acceleration

Functionality	To return the maximum acceleration of the joint.
Argument description	joint_max_acc: Return the maximum acceleration of the joint.
Command format	{s:s,s:[i,i,i,i,i,i]}
Demo	{"state":"joint_max_acc", "joint_acc":[500, 500, 500, 500, 500, 500]}
	Explain: Return the max acceleration of the 6 joints, speed: 0.5RPM/s, unit: RPM/s, resolution: 0.001RPM/s.

(3) Return joint minimum angle

Functionality	To return the minimum angle of the joint.
Argument description	joint_min_pos: Return the minimum angle of the joint.
Command format	{s:s,s:[i,i,i,i,i,i]}
Demo	{"state":"joint_min_pos", "min_pos":[-170000, -110000, -170000, -110000, -170000, -110000]}
	Explain: Return the min angles, Joint1,Joint3,Joint5:-170°, Joint2,Joint4,Joint6: -110°, unit: degree, resolution: 0.001°.

(4) Return joint maximum angle

Functionality	To return the maximum angle of the joint
Argument description	joint_max_pos: Return the max angle of the joint.
Command format	{s:s,s:[i,i,i,i,i,i]}
Demo	{"state":"joint_max_pos","max_pos":[170000,110000,170000,110000,170000,110000]}
	Explain: Return the max angles, Joint1,Joint3,Joint5:170°, Joint2,Joint4,Joint6:110°, unit: degree, resolution: 0.001°.

(5) Return the enable status of the joint

Functionality	To return the status of being enabled or not of the joint.
Argument description	joint_en_state: Return the status of being enabled or not of the joint.
Command format	{s:s,s:[i,i,i,i,i,i]}
Demo	{"state":"joint_en_state", "en_state":[1,1, 1,1,1,0]}
	Explain: Return the enable status of 6 joints, 1: enable, 0: disenable.

(6) Return the error code of the joint

Functionality	To return the error code of the joint.
Argument description	joint_err_flag: Return the error code of the joint.
Command format	{s:s,s:[i,i,i,i,i,i]}
Demo	{"state":"joint_err_flag", "err_flag":[0, 0, 0, 0, 0, 1]}
	Explain: Return the error code of 6 joints, in type of integer.

2. Robot Arm Configuration

2.1 Robot Arm Configuration-Command Set of the Movement Hyperparameters

After the arm end hyperparameters are set, they will be automatically saved to the Flash memory in the controller, and will still be valid after power off.

(1) Set the maximum line speed of the arm end

Functionality	To set the maximum line speed of the arm end.
Argument description	set_arm_max_line_speed: Set the maximum line speed of the arm end. arm_line_speed: The target line speed. Unit: m/s.
Command format	{s:s,s:i}
Demo	{"command":"set_arm_max_line_speed","arm_line_speed":500}
	Explain: Set the maximum line speed of the arm end to be 0.5m/s,resolution 0.001m/s.
Return	Format: {s:s, s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_arm_max_line_speed","arm_line_speed":true}

(2) Set the maximum line acceleration of the arm end

Functionality	To set the maximum line acceleration of the arm end.
Argument description	set_arm_max_line_acc:Set the maximum line acceleration of the arm end. arm_line_acc: The target line acceleration. Unit: m/s².
Command format	{s:s,s:i}
Demo	{"command":"set_arm_max_line_acc","arm_line_acc":2000}
	Explain: Set the maximum line acceleration of the arm end as 2 m/s², resolution: 0.001m/s².
Return	Format:{s:s, s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_arm_max_line_acc","arm_line_acc":true}

(3) Set the maximum angular speed of the arm end

Functionality	To set the maximum angular speed of the arm end.
Argument description	set_arm_max_angular_speed: Set the maximum angular speed of the arm end. arm_angular_speed: The target angular speed. Unit:rad/s
Command format	{s:s,s:i}
Demo	{"command":"set_arm_max_angular_speed","arm_angular_speed":200}

	Explain: Set the maximum angular speed of the arm end as 0.2rad/s,resolution0.001rad/s
Return	Format:{s:s, s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_arm_max_angular_speed","arm_angular_speed":200}

(4) Set the maximum angular acceleration of the arm end

Functionality	To set the maximum angular acceleration of the arm end.
Argument description	set_arm_max_angular_acc: Set the maximum angular acceleration of the arm end. arm_angular_acc: The target angular acceleration. Unit:rad/s^2
Command format	{s:s,s:i}
Demo	{"command":"set_arm_max_angular_acc","arm_angular_acc":4000}
	Explain: Set the maximum angular acceleration of the arm end as 4 rad/s^2,resolution: 0.001rad/ s^2.
Return	Format:{s:s, s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_arm_max_angular_acc","arm_angular_acc":true}

(5) Initialize the parameters of the arm end

Functionality	To initialize the parameters of the arm end.
Argument description	set_arm_init: Initialize the parameters of the arm end.
Command format	{s:s}
Demo	{"command":"set_arm_init"}
	Explain: Initialize the parameters of the arm end. The return values are restored to the default parameters of the arm end where line speed: 0.1m/s line acceleration: 0.5m/ s^2 angular speed: 0.2rad/s angular acceleration: 1 rad/ s^2
Return	Format:{s:s, s:b} ,true-Setup succeeded, false-Setup failed.
	{"command":"set_arm_init", "arm_init":true}

(6) Set controller servo opening and closing

Functionality	To set the opening and closing of the controller servo.
Argument description	set_arm_servo: Set the opening and closing of the controller servo. arm_servo: The status of the servo. 1-Open, 0-Closed.
Command format	{s:s,s:i}
Demo	{"command":"set_arm_servo", "arm_servo":1}
	Explain: By default, the controller inquires the status of the robot arm periodically after power on. In order to reduce the CANFD bus load, the servo query needs to be turned off so that this instruction is operated.
Return	Format:{s:s, s:b} ,true-Setup succeeded, false-Setup failed.
	{"command":"set_arm_servo", "arm_servo":true}

(7) Set collision protection level

Functionality	To set collision protection level/rating.
Argument description	set_collision_stage: Set collision protection level. collision_stage: Level or stage with range:0-8.
Command format	{s:s,s:i}

Demo	{"command":"set_collision_stage", "collision_stage":1}
	Explain: Set collision protection level. The higher the level, the more sensitive the detection.
Return	Format:{s:s, s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_collision_state", "collision_state":true}

(8) Query collision protection level

Functionality	To query collision protection level.
Argument description	get_collision_stage：Query the collision protection level. collision_stage: Level or range: 0~8
Command format	{s:s}
Demo	{"command":"get_collision_stage"}
	Explain: Query collision protection level. The higher the level, the more sensitive the detection.
Return	Format: {s:s, s:i}
	{"state":"get_collision_stage","collision_stage":5}

(9) Reset the DH hyperparameters

Functionality	To reset the DH hyperparameters.
Argument description	set_DH_data: Reset the DH hyperparameter.
Command format	{s:s,s:[i, i, i, i, i]}
Demo	{"command":"set_DH_data", "data":[2405, 2560, 2100, 1440, 0]}
	Explain: Reset the DH hyperparameter. Precision: 0.1mm. The content of the above command contains lsb: 240.5mm lse: 256mm lew: 210mm lwt: 144mm d3: 0mm
Return	Format:{s:s, s:b},true-Setup succeeded, false-Setup failed.
	{"command":"set_DH_data ", "set_state":true}
Comment	This command cannot be used by the user own self, and must be used only when the absolute accuracy is compensated with the measuring equipment, otherwise it will lead to wrong parameters of the robot arm.

(10) Inquiry of robot DH hyperparameters

Functionality	To inquire DH hyperparameters.
Argument description	get_DH_data：Inquire robot DH hyperparameters.
Command format	{s:s,s:i}
Demo	{"command":"get_DH_data"}
	Explain：Inquire robot DH hyperparameters.
Return	Format：{s:s,s:[i,i,i,i,i]}
	{"state":"DH_data","data":[2405,2560,2100,1440,0]}
	Explain : Set DH hyperparameters , precision : 0.1mm. The above

	command contents are as follows： lsb:240.5mm lse:256mm lew:210mm lwt:144mm d3:0mm

(11) Reset the compensation angle to the joint origin

Functionality	To reset the compensation angle to the joint origin, which is used to correct the absolute positioning accuracy.
Argument description	set_joint_zero_offset
Command format	{s:s,s:[i, i, i, i, i, i]}
Demo	{"command":"set_joint_zero_offset", "offset":[1000, -2000, 3000, -4000, 5000, -6000]}
	Explain: Set the compensation angle of the joint origin. Resolution: 0.001°. The compensation angle of Joint1~Joint6: 1°,-2°,3°,-4°,5°,-6°.
Return	Format:{s:s, s:b} ,true-Setup succeeded, false-Setup failed.
	{"command":"set_joint_zero_offset", "set_state":true}
Comment	This command cannot be used by the user own self, and must be used only when the absolute accuracy is compensated with the measuring equipment, otherwise it will lead to wrong parameters of the robot arm.

(12) Reset the robot arm dynamics parameters

Functionality	To reset the robot arm dynamics parameters.
Argument description	set_arm_dynamic_parm
Command format	{s:s,s:[i, i, i, i, i, i , i, i, i, i, i, i]}
Demo	{"command":"set_arm_dynamic_parm", "parm":[1000, -2000, 3000, -4000, 5000, -6000, 1000, -2000, 3000, -4000, 5000, -6000]}
	Explain: Set the robot arm dynamics parameters, accuracy: 0.001
Return	Format: {s:s, s:b} , true- Setup succeeded, false-Setup failed.
	{"command":"set_arm_dynamic_parm", "set_state":true}

2.2 Robot Arm Configuration-the Query Set of Movement Parameters

(1) Query the maximum linear speed of the arm end

Functionality	To query the maximum linear speed of the arm end.
Argument description	get_arm_max_line_speed: Query the maximum linear speed of the arm end.
Command format	{s:s}
Demo	{"command":"get_arm_max_line_speed"}
	Explain: Query the maximum linear speed of the arm end.

Return	See Table 2.3-(1)

(2) Query the maximum linear acceleration of the arm end

Functionality	To query the maximum linear acceleration of the arm end.
Argument description	get_arm_max_line_acc: Query the maximum linear acceleration of the arm end.
Command format	{s:s}
Demo	{"command":"get_arm_max_line_acc"}
	Explain: Query the maximum linear acceleration of the arm end.
Return	See Table 2.3-(2)

(3) Query the maximum angular speed of the arm end

Functionality	To query the maximum angular speed of the arm end.
Argument description	get_arm_max_angular_speed: Query the maximum angular speed of the arm end.
Command format	{s:s}
Demo	{"command":"get_arm_max_angular_speed"}
	Explain: Query the maximum angular speed of the arm end.
Return	See Table 2.3-(3)

(4) Query the maximum angular acceleration of the arm end

Functionality	To query the maximum angular acceleration of the arm end.
Argument description	get_arm_max_angular_acc: Query the maximum angular acceleration of the arm end.
Command format	{s:s}
Demo	{"command":"get_arm_max_angular_acc"}
	Explain: Query the maximum angular acceleration of the arm end.
Return	See Table 2.3-(4)

2.3 Robot Arm Configuration-Return Movement Parameters

(1) Return the maximum linear speed of the arm end

Functionality	To return the maximum linear speed of the arm end.
Argument description	arm_max_line_speed: Return the maximum linear speed of the arm end.
Command format	{s:s,s:i}
Demo	{"state":"arm_max_line_speed", "arm_line_speed":500
	Explain: Return the maximum linear speed of the arm end, 0.5m/s, resolution:0.001m/s

(2) Return the maximum linear acceleration of the arm end

Functionality	To return the maximum linear acceleration of the arm end.

Argument description	arm_max_line_acc: Return the maximum linear acceleration of the arm end.
Command format	{s:s,s:i}
Demo	{"state":"arm_max_line_acc", "arm_line_acc":200
	Explain: Return the maximum linear acceleration of the arm end, 0.2m/s², resolution:0.001 m/s²

(3) Return the maximum angular speed of the arm end

Functionality	To return the maximum angular speed of the arm end.
Argument description	arm_max_angular_speed: Return the maximum angular speed of the arm end.
Command format	{s:s,s:i}
Demo	{"state":"arm_max_angular_speed", "arm_angular_speed":1000}
	Explain: Return the maximum angular speed of the arm end, 1rad/s, resolution:0.001rad/s

(4) Return the maximum angular acceleration of the arm end

Functionality	To return the maximum angular acceleration of the arm end.
Argument description	arm_max_angular_acc: Return the maximum angular acceleration of the arm end.
Command format	{s:s,s:i}
Demo	{"state":"arm_max_angular_acc", "arm_angular_acc":10000}
	Explain: Return the maximum angular acceleration of the arm end,10rad/s², resolution:0.001rad/ s²

2.4 Robot Arm Configuration-the Command Set of the Tool End

(1) Automatically calculate the coordinate system of the tool end (Calibrate the reference points)

Functionality	Automatic calculation of the coordinate system of the tool end (using the six-point method): The robot can only store 10 tool-end coordinate systems. If there are more than 10, the new tool will not succeed.
Argument description	set_auto_tool_frame: Automatically calculate the tool-end coordinate system. tool_name: Name of the tool-end coordinate system. Must be less 10 characters. point_num:1~6 are the reference points for calibration,7 is the tool for automatic calculation.
Command format	{s:s,s:s,s:i}

Demo	{"command":"set_auto_tool_frame","point_num":1}
	{"command":"set_auto_tool_frame","point_num":2}
	{"command":"set_auto_tool_frame","point_num":3}
	{"command":"set_auto_tool_frame","point_num":4}
	{"command":"set_auto_tool_frame","point_num":5}
	{"command":"set_auto_tool_frame","point_num":6}

	Explain: Automatically calculate the tool-end coordinate system; name:

	tool2_frame; set the current position as the reference point6. Note: After the robot arm is initialized when power on, there is no load by default.
Return	Format:{s:s, s:b}　,true-Setup succeeded,　false-Setup failed.
	{"command":"set_auto_tool_frame","auto_tool_frame":true}

(2) Automatically calculate tool coordinate system (automatic calculation generation tool)

Functionality	Automatic calculation of tool coordinate system (six-point method): the robot arm can only store 10 tool coordinate systems. If more than 10, the new tool creation will not be successful.
Argument description	set_auto_tool_frame：Automatically compute the tool coordinate system. tool_name： Tool coordinate system name, which cannot exceed 10 characters. payload: Unit： g， no more than 5000g； position：Position of the center of mass. Unit：mm，precision: 0.001mm
Command format	{s:s,s:s,s:i,s:[i,i,i]}
Demo	{"command":"generate_auto_tool_frame","tool_name":"tool2_frame", "payload":5000,"position":[1000,2000,3000]}
	Explain： Automatically calculate the tool coordinate system whose name is tool2_frame. Mark the current position as a reference point 6. End payload is 5000g. Position of the center of mass ： x-1mm,y-2mm,z-3mm Comment： After the robot arm is initialized when power on, there is no load by default.
Return	Format： {s:s,s:b}， true-Setup succeeded,　false-Setup failed.
	{"command":"set_auto_tool_frame","auto_tool_frame":true}

(3) Manually input the coordinate system of the tool end

Functionality	To manually input the tool-end coordinate system: The robot can only store 10 tool-end coordinate systems. If there are more than 10, the new tool will not succeed.
Argument description	set_manual_tool_frame: Manually input the hyperparameters of the tool-end coordinate system. tool_name: Name of the tool-end coordinate system. Must be less 10 characters. tool_pose: Tool position and pose relative to the center of the flange of the arm end.
Command format	{s:s,s:s,s:[i,i,i,i,i,i],s:i,s:[i,i,i]}
Demo	{"command":"set_manual_tool_frame", "tool_name":"tool2_frame", "tool_pose":[100000, 200000, 30000, 400, 500, 600],"payload":5000,"position":[1000,2000,3000]}
	Explain: Manually input the tool-end coordinate system; Name: tool2_frame. Tool position: x:0.1m,y:0.2m,z:0.03m,position resolution:0.001mm Tool pose: rx:0.4rad,ry:0.5rad,rz:0.6rad,pose resolution:0.001rad

	End payload: 5000g , position of the center of mass : x-1mm,y-2mm,z-3mm payload: Unit：g，no more than 5000g； position：Position of the center of mass. Unit：mm，precision: 0.001mm Comment： After the robot arm is initialized when power on, there is no load by default.
Return	Format:{s:s,s:b} , true-Setup succeeded, false-Setup failed.
	{"command":"set_manual_tool_frame", "manual_tool_frame":true}

(4) Switch the current tool-end coordinate system

Functionality	To switch the current tool-end coordinate system.
Argument description	set_change_tool_frame: Switch the current tool-end coordinate system. tool_name: Name of the tool-end coordinate system.
Command format	{s:s,s:s}
Demo	{"command":"set_change_tool_frame", "tool_name":"tool2_frame"}
	Explain: Switch the current tool-end coordinate system. Name: tool2_frame.
Return	Format:{s:s,s:b} , true-Setup succeeded, false-Setup failed.
	{"command":"set_change_tool_frame", "change_tool_name":true}

(5) Delete a tool-end coordinate system

Functionality	To delete a tool-end coordinate system.
Argument description	set_delete_tool_frame: Delete a tool-end coordinate system. tool_name: Name of the tool-end coordinate system.
Command format	{s:s,s:s}
Demo	{"command":"set_delete_tool_frame", "tool_name":"tool2_frame"}
	Explain: Delete a tool-end coordinate system. Name: tool2_frame .
Return	Format:{s:s,s:b} , true-Setup succeeded, false-Setup failed.
	{"command":"set_delete_tool_frame", "delete_tool_name":true}

2.5 Robot Arm Configuration-the Command Set of Operation Coordinate System

(1) Automatically set the operation coordinate system

Functionality	To set the operating or working coordinate system. The robot can only store 10 operation coordinate systems. If there are more than 10, the new tool will not succeed.
Argument description	set_work_frame: Set the operating or working coordinate system. frame_name: Name of the operation coordinate system. Must be less 10 characters. point_num: The reference point1~3 denote the origin of the operation coordinate system、a point on x axis and a point on y axis, respectively. Point 4 denotes the operation coordinate system that is calibrated by the prior 3 points.
Command	{s:s,s:s,s:i}

format	
Demo	{"command":"set_auto_work_frame","frame_name":"work2_frame", "point_num":3}
	Explain: Set the operation coordinate system, name: work2_frame, set the current position as Point 3.
Return	Format:{s:s,s:b} , true-Setup succeeded, false-Setup failed.
	{"command":"set_auto_work_frame","auto_work_frame":true}

(2) Manually input the operation coordinate system

Functionality	To manually input the operation coordinate system. The robot can only store 10 operation coordinate systems. If there are more than 10, the new tool will not succeed.
Argument description	set_manual_work_frame: Manually input the operation coordinate system. frame_name: Name of the operation coordinate system. Must be less 10 characters. frame_pose: Operating position.
Command format	{s:s,s:s,s:[i,i,i,i,i,i]}
Demo	{"command":"set_manual_work_frame","frame_name":"work2_frame", "frame_pose":[100000, 200000, 30000, 400, 500, 600]}
	Explain: Manually input the operation coordinate system. Name: work2_frame, Location of the coordinate system: x:0.1m,y:0.2m,z:0.03m,position resolution: 0.001mm Pose of the coordinate system: rx:0.4rad,ry:0.5rad,rz:0.6rad,pose resolution: 0.001rad
Return	Format:{s:s,s:b} , true-Setup succeeded, false-Setup failed.
	{"command":"set_manual_work_frame", "manual_work_frame":true}

(3) Switch the current operation coordinate system

Functionality	To switch the current operation coordinate system.
Argument description	set_change_work_frame: Switch the current operation coordinate system. frame_name: Name of the coordinate system.
Command format	{s:s,s:s}
Demo	{"command":"set_change_work_frame","frame_name":"work2_frame"}
	Explain: Switch the current operation coordinate system. Name: work2_frame.
Return	Format:{s:s,s:b} , true-Setup succeeded, false-Setup failed.
	{"command":"set_change_work_frame","change_work_frame":true}

(4) Delete an operation coordinate system

Functionality	To delete an operation coordinate system.
Argument description	set_delete_work_frame: Delete an operation coordinate system. frame_name: Name of the coordinate system.
Command format	{s:s,s:s}

Demo	{"command":"set_delete_work_frame","frame_name":"work2_frame"}
	Explain: Delete an operation coordinate system. Name: work2_frame .
Return	Format:{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_delete_work_frame","delete_work_frame":true }

2.6 Robot Arm Configuration-the Query Set of the Coordinate System

(1) Query the current coordinate system

Functionality	To query the current coordinate system.
Argument description	get_current_tool_frame: Query the current coordinate system.
Command format	{s:s}
Demo	{"command":"get_current_tool_frame"}
	Explain: Query the current coordinate system.
Return	See Table 2.7-(1)

(2) Query all coordinate system(s)

Functionality	To query all coordinate system(s).
Argument description	get_total_tool_frame: Query the name of all coordinate system(s).
Command format	{s:s}
Demo	{"command":"get_total_tool_frame"}
	Explain: Query the names of all coordinate system(s).
Return	See Table 2.7-(2)

(3) Query the specified tool information

Functionality	To query the specified tool information.
Argument description	get_tool_frame: Query the specified tool information. tool_name: The tool name.
Command format	{s:s, s:s}
Demo	{"command":"get_tool_frame", "tool_name":"tool"}
	Explain: Query the specified tool information. Name: tool.
Return	See Table 2.7-(3)

(4) Query the current operation coordinate system

Functionality	To query the current operation coordinate system.
Argument description	get_current_work_frame: Query the current operation coordinate system.
Command format	{s:s}
Demo	{"command":"get_current_work_frame"}
	Explain: Query the current operation coordinate system.
Return	See Table 2.7-(4)

(5) Query the name(s) of the existed coordinate system(s)

Functionality	To query the names of the existed coordinate system(s).
Argument description	get_total_work_frame: Query the names of the existed coordinate system(s).
Command format	{s:s}
Demo	{"command":"get_total_work_frame"}
	Explain: Query the names of the existed coordinate system(s).
Return	See Table 2.7-(5)

(6) Query the specified coordinate system

Functionality	To query the specified coordinate system.
Argument description	get_work_frame: Query the specified coordinate system. frame_name: The coordinate system name.
Command format	{s:s,s:s}
Demo	{"command":"get_work_frame", "frame_name":"work1"}
	Explain: Query the specified coordinate system. Name: work1
Return	See Table 2.7-(6)

2.7 Robot Arm Configuration-the Return Set of the Coordinate System(s)

(1) Return the current tool information

Functionality	To return the current tool information.
Argument description	current_tool_frame: Return the current tool information.
Command format	{s:s,s:s,s:[i,i,i,i,i,i],s:i,s:[i,i,i]}
Demo	{"state":"current_tool_frame", "tool_name":"tool2_frame", "pose":[100000, 200000, 30000, 400, 500, 600],"payload":5000,"position":[1000,2000,3000]}
	Explain: Return the current tool information. Tool name: tool2_frame Tool position: x:0.1m,y:0.2m,z:0.03m. Position resolution: 0.001mm Tool pose: rx:0.4rad,ry:0.5rad,rz:0.6rad. Pose resolution: 0.001rad Payload：5kg, precision: 0.001kg Position of the center of mass：1mm, precision: 0.001mm

(2) Return the name(s) of the existed tool(s)

Functionality	To return the name(s) of the tool(s). Return NULL if empty.
Argument description	total_tool_frame: Return the name(s) of the tool(s).
Command format	{s:s,s:s,s:[s,s,…,s]}
Demo	{"state":"total_tool_frame","tool_names":["base_tool1", "base_tool2"….,"NULL"]}
	Explain: Return the names of 10 tools. Tool names: base_tool1,base_tool2,…, base_tool10. “NULL” denotes an unestablished coordinate system.

(3) Return the specified tool information

Functionality	To return the specified tool information.
Argument description	given_tool_frame: Return the specified tool information.
Command format	{s:s,s:s,s:[i,i,i,i,i,i],s:i,s:[i,i,i]}
Demo	{"state":"given_tool_frame","tool_name":"tool2_frame","pose":[100000,200000,30000,400,500,600],"payload":5000,"position":[1000,2000,3000]}
	Explain: Return the specified tool information. Tool name: tool2_frame, Tool position: x:0.1m,y:0.2m,z:0.03m,Position resolution: 0.001mm Payload：5kg, precision: 0.001kg Position of the center of mass：1mm, precision: 0.001mm Tool pose: rx:0.4rad,ry:0.5rad,rz:0.6rad,Pose resolution: 0.001rad

(4) Return the information of the current operation coordinate system

Functionality	To return the information of the current operation coordinate system.
Argument description	current_work_frame: Return the information of the current operation coordinate system.
Command format	{s:s,s:s,s:[i,i,i,i,i,i]}
Demo	{"state":"current_work_frame", "frame_name":"work2_frame","pose":[100000, 200000, 30000, 400, 500, 600]}
	Explain: Return the information of the current operation coordinate system Name: work2_frame, Coordinate system position: x:0.1m,y:0.2m,z:0.03m. Position resolution: 0.001mm Coordinate system pose: rx:0.4rad,ry:0.5rad,rz:0.6rad,Pose resolution: 0.001rad

(5) Return the name(s) of the existed coordinate system(s)

(6) Return the information of the specified coordinate system

Functionality	To return the information of the specified coordinate system.
Argument description	given_work_frame: Return the information of the specified coordinate system.
Command format	{s:s,s:s,s:[i,i,i,i,i,i]}
Demo	{"state":"given_work_frame","frame_name":"work2_frame", "pose":[100000, 200000, 30000, 400, 500, 600]}
	Explain: Return the information of the specified coordinate system. Name of the specified coordinate system: work2_frame. Coordinate system position: x:0.1m,y:0.2m,z:0.03m,resolution: 0.001mm Coordinate system pose: rx:0.4rad,ry:0.5rad,rz:0.6rad,resolution:

	0.001rad

2.8 Robot Arm Configuration- the Query Set of the Status

(1) Query the status of the robot arm

Functionality	To query the status of the robot arm
Argument description	get_current_arm_state: Query the status of the robot arm.
Command format	{s:s}
Demo	{"command":"get_current_arm_state"}
	Explain: Query the status of the robot arm.
Return	See Table 2.9-(1)

(2) Query the temperature of the joint

Functionality	To query the temperature of the joint.
Argument description	get_current_joint_temperature: Query the temperature of the joint.
Command format	{s:s}
Demo	{"command":"get_current_joint_temperature"}
	Explain: Query the temperature of the joint.
Return	See Table 2.9-(2)

(3) Query the current of the joint

Functionality	To query the current of the joint.
Argument description	get_current_joint_current: Query the current of the joint.
Command format	{s:s}
Demo	{"command":"get_current_joint_current"}
	Explain: Query the current of the joint.
Return	See Table 2.9-(3)

(4) Query the voltage of the joint

Functionality	To query the voltage of the joint.
Argument description	get_current_joint_voltage: Query the voltage of the joint.
Command format	{s:s}
Demo	{"command":"get_current_joint_voltage"}
	Explain: Query the voltage of the joint.
Return	See Table 2.9-(4)

2.9 Robot Arm Configuration-the Return Set of the Status

(1) Return the status of the robot arm

Functionality	To return the status of the robot arm, including the joint angles, arm-end position and pose, the error code of the arm and the error code of the controller.

Argument description	current_arm_state: Return the status of the robot arm. joint:joint angles pose:arm-end position and pose arm_err:the error code of the arm sys_err:the error code of the controller
Command format	{s:s,s:{s:[i,i,i,i,i,i],s:[i,i,i,i,i,i],s:i,s:i}}
Demo	{"state":"current_arm_state", "arm_state":{"joint":[100,200,300,400,500,600], "pose":[100000, 200000, 30000, 400, 500, 600], "arm_err":0, "sys_err":0}}
	Explain: Return the status of the robot arm. The angles of Joint1~Joint6:0.1°,0.2°,0.3° 。0.4°,0.5°,0.6°,resolution:0.001° Position:x:0.1m,y:0.2m,z:0.03m,resolution:0.001mm Pose:rx:0.4rad,ry:0.5rad,rz:0.6rsd,resolution:0.001rad The error code of the arm, referring to a software error in the operation of the arm:0 The error code of the controller, referring to a hardware error in the operation of the arm:0

(2) Return the temperature of the joint

Functionality	To return the temperature of the joint.
Argument description	current_joint_temperature: Return the temperature of the joint. Unit:℃
Command format	{s:s,s:[i,i,i,i,i,i]}
Demo	{"state":"current_joint_temperature", "joint_temperature":[27500, 28000, 26800, 26800, 28900, 30100]}
	Explain: Return the temperature of the joint. Temperatures of the joints: [27.5, 28.0, 26.8, 26.8, 28.9, 30.1],unit:℃,resolution:0.001℃

(3) Return the current of the joint

Functionality	To return the current of the joint.
Argument description	current_joint_current: Return the current of the joint. Unit: mA. Resolution:0.001mA
Command format	{s:s,s:[i,i,i,i,i,i]}
Demo	{"state":"current_joint_current", "joint_current":[65,-200, 170, 200, -300, 168]}
	Explain:Return the current of the joint. The currents of Joint1~Joint6:0.065mA,-0.2mA, 0.17mA, 0.2mA, -0.3mA, 0.168mA

(4) Return the voltage of the joint

Functionality	To return the voltage of the joint.
Argument description	current_joint_voltage: Return the voltage of the joint. Unit:V. Resolution:0.001V
Command format	{s:s, s:[i,i,i,i,i,i]}
Demo	{"state":"current_joint_voltage", "joint_voltage":[27500, 28000, 26800, 26800, 28900, 30100]}

	Explain: Return the voltage of the joint. The voltages of Joint1~Joint6: 27.5V, 28.0V, 26.8V, 26.8V, 28.9V, 30.1V

(5) Return the system error(s) of the robot arm

Functionality	To return the system error(s) of the robot arm.
Argument description	current_arm_err: Return the system error(s) of the robot arm.
Command format	{s:s,s:i}
Demo	{"state":"current_arm_err", "arm_err":8}
	Explain: Return the system error(s) of the robot arm. Error code: 8

2.10 Robot Arm Configuration-Initial Pose

(1) Set the initial poses

Functionality	To set the initial poses.
Argument description	set_init_pose:Set the initial poses. init_pose:Set the initial poses. Resolution: 0.001°
Command format	{s:s,s:[i, i, i, i, i, i]}
Demo	{"command":"set_init_pose", "init_pose":[10000, 0, 20000, 30000, 0, 20000]}
	Explain: Set the initial poses. Initial poses: [10°,0°,20°,30°,0°,20°]
Return	Format:{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_init_pose", "init_pose":true}

(2) Query the initial poses

Functionality	To query the initial poses.
Argument description	get_init_pose:Query the initial poses.
Command format	{s:s}
Demo	{"command":"get_init_pose"}
	Explain: Query the initial poses.
Return	See the table as below（3）

(3) Return the initial poses

Functionality	To return the initial poses.
Argument description	init_pose:Return the initial poses. init_pose:resolution 0.001°
Command format	{s:s,s:[i,i,i,i,i,i]}
Demo	{"state":"init_pose", "init_pose":[10000, 0, 20000, 30000, 0, 20000]}
	Explain: Return the initial poses. Initial pose: [10°,0°,20°,30°,0°,20°]

3. Movement Configuration

3.1 Movement Configuration-the Command Set of Trajectory

(1) MoveJ:Joint movement

Functionality	MoveJ:Joint movement.
Argument	movej:Joint movement.

description	joint:Target joint angle,resolution 0.001° v:Speed percentage value,0~100 r:blending radius,resolution 0.001m. Blending mode is not supported for now with a default value: 0.
Command format	{s:s,s:[i,i,i,i,i,i],s:i,s:i}
Demo	{"command":"movej", "joint":[10100, 200, 20300, 30400, 500, 20600], "v":50, "r":0}
	Explain: Joint movement. Joint angles [10.1°, 0.2°, 20.3°, 30.4°, 0.5°, 20.6°]. Speed percentage 50%. Blending radius:0
Return	Format:{s:s,s:b}, true-Reached the target, false-Planning failed
	{"state":"current_trajectory_state", "trajectory_state":true}

(2) MoveL:Linear movement

Functionality	MoveL:Linear movement
Argument description	movel:Linear movement pose:The target pose,position resolution:0.001mm,pose resolution:0.001rad v:Speed percentage value,0~100 r:Blending radius,resolution 0.001m. Blending mode is not supported for now with a default value: 0.
Command format	{s:s,s:[i,i,i,i,i,i],s:i,s:i}
Demo	{"command":"movel", "pose":[100000, 200000, 30000, 400, 500, 600], "v":50, "r":0}
	Explain: Linear movement, Target position:x:0.1m,y:0.2m,z:0.03m Target pose:rx:0.4rad,ry:0.5rad,rz:0.6rad Speed percentage value 50% Non-blending mode.
Return	Format:{s:s,s:b}, true- Reached the target, false-Planning failed
	{"state":"current_trajectory_state", "trajectory_state":true}
Comment	MOVL is also applicable to the pose change with the target position unchanged

(3)MoveC: Circular movement

Functionality	MoveC:Circular movement
Argument description	movec:Circular movement pose:position and pose pose_via:Position and pose of the middle point,position resolution 0.001mm,pose resolution 0.001rad pose_to:Target position and pose,position resolution 0.001mm,pose resolution 0.001rad v:Speed percentage value,0~100 r:Blending radius. Blending mode is not supported for now with a default value: 0. loop:Number of loops, by default 0
Command format	{s:s, s:{s:[i,i,i,i,i,i], s:[i,i,i,i,i,i]},s:i,s:i, s:i}

Demo	{"command":"movec", "pose":{"pose_via": [100000, 200000, 30000, 400, 500, 600], "pose_to":[200000, 300000, 30000, 400, 500, 600]}, "v":50, "r":0, "loop":0}
	Explain: Circular movement, Position of the middle point:x:0.1m,y:0.2m,z:0.03m Pose of the middle point:rx:0.4rad,ry:0.5rad,rz:0.6rad Final position:x:0.2m,y:0.3m,z:0.03m Final pose:rx:0.4rad,ry:0.5rad,rz:0.6rad Speed percentage 50%, Non-blending No loop.
Return	Format:{s:s,s:b}, true- Reached the target, false-Planning failed
	{"state":"current_trajectory_state", "trajectory_state":true}

(4) Angle Passthrough via CANFD

Functionality	movej_canfd:To passthrough/transmit angles via CANFD. No use of the controller.
Argument description	movej_canfd: Passthrough angles via CANFD. If succeeded, the robot arm proceeds immediately. joint:The joint angles, resolution 0.001° Note: The faster the transmission period, the better and smoother the control effect. 20ms is the fastest transmission period for WIFI and network port mode, 10ms for USB and RS485 mode. 10ms is also the fastest transmission period for high-speed network port, but you need to use the command to open the configuration before using the high-speed network port.
Command format	{s:s, s:[i,i,i,i,i,i]}
Demo	{"command":"movej_canfd", "joint":[1000, 0, 20000, 30000, 0, 20000]}
	Explain: Transmit angles via CANFD. Target angles:[1°, 0°, 20°, 30°, 0, 20°]
Return	Format: {s:s,s: [i,i,i,i,i,i],s:i},
	{"state":"joint_state ","joint":[10,20,30,40,50,60], "arm_err":0} Joint precision： 0.001° arm_err： If it is 0, it means the system is normal and the command runs normally. If it is other error, the corresponding error code will be fed back and the command will not be executed.

(5) Pose Passthrough

Functionality	movep_canfd： The target position is transmitted to the robot arm without the need for controller planning.
Argument description	movep_canfd： After the target pose is transmitted to the robot arm, the controller performs the inverse solution, and if the inverse solution exists and the inverse solution does not have a large difference between the angles and the current angle, it is directly sent to the joint for execution without further trajectory planning.

	It is suitable for scenarios where the user needs to adjust the pose periodically, such as visual servo, etc. Note: The faster the transmission period, the better and smoother the control effect. 20ms is the fastest transmission period for WIFI and network port mode, 10ms for USB and RS485 mode. 10ms is also the fastest transmission period for high-speed network port, but you need to use the command to open the configuration before using the high-speed network port.
Command format	{s:s,s:[i,i,i,i,i,i]}
Demo	{"command":"movep_canfd","pose":[100000,200000,30000,400,500,600]}
	Explain：pose：Target pose，Position precision：0.001mm，Posture precision：0.001rad Target position：x： 0.1m， y:0.2m， z： 0.03m Target posture： rx:0.4rad， ry:0.5rad， rz:0.6rad The target pose is the value of the current tool in the current working/operation coordinate system.
Return	Format： {s:s,s: [i,i,i,i,i,i],s:i}
	{"state":"pose_state ","pose":[10,20,30,40,50,60], "arm_err":0} pose： current pose， position precision： 0.001mm， posture precision：0.001rad arm_err： If it is 0, it means the system is normal and the command runs normally. If it is other error, the corresponding error code will be fed back and the command will not be executed.

(6) MoveJ_P: Joint space planning to target pose and position

Functionality	MoveJ_P: Joint space planning to target pose and position.
Argument description	movej_p: Joint space planning to target pose and position. pose: Target position and pose， position accuracy: 0.001mm， pose accuracy: 0.001rad v: Speed percentage, 0~100 r: blending radius, resolution 0.001m. Currently not support blending with default value of 0
Command format	{s:s,s:[i,i,i,i,i,i],s:i,s:i}
Demo	{"command":"movej_p", "pose":[100000, 200000, 30000, 400, 500, 600], "v":50, "r":0}
	Comment: Linear movement, Target position: x:0.1m, y:0.2m, z:0.03m, Target pose: rx:0.4rad, ry:0.5rad, rz:0.6rad, Speed percentage 50%, No blending.
Return	Format: {s:s,s:b}, true-Successful reach to the target pose and position, false-Planning failure.
	{"state":"current_trajectory_state", "trajectory_state":true}
Comment	The target pose and position must be the pose and position of the flange

	center of the arm end based on the base coordinate system. The user must ensure it before using this command, otherwise the target pose and position will encounter errors.

3.2 Movement Configuration-the Command Set of the Step

(1) Set the stepping joint

Functionality	To set the stepping joint.
Argument description	set_joint_step:Set the stepping joint. joint_step:（1）Joint #; （2）Step angles,unit:°,resolution:0.001° v:Speed percentage value,0~100
Command format	{s:s,s:[i,i]}
Demo	{"command":"set_joint_step", "joint_step":[1, -10000], "v":30}
	Explain: Set the joint step. Joint 1 rotates 10° to the opposite direction. Speed percentage value: 30%.
Return	Format:{s:s,s:b}, true-Reach the target location, false-Planning failed.
	{"state":"current_trajectory_state", "trajectory_state":true}

(2) Set the stepping position

Functionality	To set the stepping position.
Argument description	set_pos_step:Set the stepping position. step_type:The step type,x_step: x-axis direction,y_step: y-axis direction,z_step: z-axis direction. step: Step distance. Unit: m, resolution:0.001mm, i.e., 0.000001m v: Speed percentage value.
Command format	{s:s,s:s,s:i,s:i}
Demo	{"command":"set_pos_step", "step_type":"x_step", "step":-50000, "v":30}
	Explain: Move the position by 0.5m in negative x-axis direction with speed of 30%.
Return	Format:{s:s,s:b}, true-Reach the target location, false-Planning failed.
	{"state":"current_trajectory_state", "trajectory_state":true}

(3) Set the stepping pose

Functionality	To set the stepping pose
Argument description	set_ort_step:Set the pose. step_type:The step type,rx_step: rotate along x axis,ry_step:rotate along y axis, rz_step:rotate along z axis step:Stepping arc,unit: rad,resolution: 0.001rad v:Speed percentage value.
Command format	{s:s,s:s,s:i,s:i}
Demo	{"command":"set_ort_step", "step_type":"rx_step", "step":-500,

	"v":30}
	Explain: Set the stepping pose of 0.5rad in negative x axis with speed of 30%.
Return	Format:{s:s,s:b}, true- Reach the target location, false-Planning failed.
	{"state":"current_trajectory_state", "trajectory_state":true}

3.3 Movement Configuration-the Command Set of Motion

(1) Terminate the trajectory

Functionality	To terminate the trajectory.
Argument description	set_arm_stop:Terminate the ongoing trajectory with the shortest time. The trajectory is not recoverable.
Command format	{s:s}
Demo	{"command":"set_arm_stop"}
	Explain: Terminate the trajectory immediately.
Return	Format:{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_arm_stop", "arm_stop":true}

(2) Pause the trajectory

Functionality	To pause the trajectory.
Argument description	set_arm_pause:Pause the trajectory during operation. The trajectory is recoverable.
Command format	{s:s}
Demo	{"command":"set_arm_pause"}
	Explain: Pause the trajectory.
Return	Format:{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_arm_pause", "arm_pause":true}

(3) Continue the trajectory

Functionality	To continue the trajectory.
Argument description	set_arm_continue:Continue the paused trajectory.
Command format	{s:s}
Demo	{"command":"set_arm_continue"}
	Explain: Continue the paused trajectory.
Return	Format:{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_arm_continue", "arm_continue":true}

(4) Delete the current trajectory

Functionality	To delete the current trajectory. Must pause the trajectory before use.
Argument description	set_delete_current_trajectory:Delete the current trajectory.
Command format	{s:s}
Demo	{"command":"set_delete_current_trajectory"}

	Explain: Delete the current trajectory.
Return	Format:{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_arm_delete_current_trajectory","delete_current_trajectory ":true}

(5) Delete all trajectory

Functionality	To delete all trajectory. Must pause the trajectory before use.
Argument description	set_arm_delete_trajectory:Delete all trajectory.
Command format	{s:s}
Demo	{"command":"set_arm_delete_trajectory"}
	Explain: Delete all trajectory.
Return	Format:{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_arm_delete_trajectory","arm_delete_trajectory ":true}

(6) Query the current planning type

Functionality	To query the current planning type.
Argument description	get_arm_current_trajectory:Query the current trajectory.
Command format	{s:s}
Demo	{"command":"get_arm_current_trajectory"}
	Explain: Query the current trajectory.
Return	See Table 3.5-(1)

3.4 Movement Configuration-the Command Set of Teaching

(1) Teach the joint

Functionality	To teach the joint.
Argument description	set_joint_teach:Teach the joint. teach_joint:Joint #. direction:Direction,“pos”: positive direction,“neg”:negative direction. v:Speed percentage value.
Command format	{s:s,s:i,s:s,s:i}
Demo	{"command":"set_joint_teach","teach_joint":1,"direction":"pos","v":50}
	Explain: Teach the Joint 1,positive direction,speed 50%
Return	Format:{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":" set_joint_teach","joint_teach":true}

(2) Teach the position

Functionality	To teach the position.
Argument description	set_pos_teach:Teach the position. teach_type:Coordinate system,”x”, ”y”, ”z” direction:Direction,“pos”: positive direction,“neg”:negative direction. v:Speed percentage value.

Command format	{s:s,s:s,s:s,s:i}
Demo	{"command":"set_pos_teach", "teach_type":"x", "direction":"neg", "v":50}
	Explain: Teach the position,negative x axis,speed 50%
Return	Format:{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_pos_teach","pos_teach":true}

(3) Teach the pose

Functionality	To teach the pose
Argument description	set_ort_teach:Teach the pose. teach_type:Rotation axis,“rx”, “ry”, “rz”. direction:Direction,“pos”: positive direction,“neg”:negative direction. v:Speed percentage value.
Command format	{s:s,s:s,s:s,s:i}
Demo	{"command":"set_ort_teach", "teach_type":"rx", "direction":"neg", "v":50}
	Explain: Teach the pose,negative rx axis,speed 50%
Return	Format:{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_ort_teach","ort_teach":true}

(4) Stop the teach

Functionality	To stop the teach.
Argument description	set_stop_teach:Stop the teach.
Command format	{s:s}
Demo	{"command":"set_stop_teach"}
	Explain: Stop the teach.
Return	Format:{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_stop_teach","stop_teach":true}

3.5 Movement Configuration-the Return Set of the Trajectory

(1) Return the current ongoing trajectory

Functionality	To return the current ongoing trajectory.
Argument description	arm_current_trajectory:Return the current ongoing trajectory.
Command format	{s:s, s:s, s:[i,i,i,i,i,i]}
Demo	{"state":"arm_current_trajectory", "type":"movej", "data":[0, 0, 0, 0, 0, 0]} Explain: Get the planning state of the current ongoing trajectory. Current angles of 6 joints are included in the array, resolution: 0.001°.
	{"state":"arm_current_trajectory", "type":"movel", "data":[0, 0, 0, 0, 0, 0]} Explain: The current ongoing planning is linear movement. The array shows the current position and pose of the arm end. Position resolution:0.001mm,pose resolution:0.001rad

	{"state":"arm_current_trajectory", "type":"movec", "data":[0, 0, 0, 0, 0, 0]} Explain: The current ongoing planning is circular movement. The array shows the current position and pose of the arm end. Position resolution:0.001mm,pose resolution:0.001rad
	{"state":"arm_current_trajectory", "type":"none", "data":[0, 0, 0, 0, 0, 0]} Explain: No ongoing planning. The array shows the current angles of 6 joints. Angle resolution: 0.001°

(2) Return the accomplishment status of the current trajectory

Functionality	To return the accomplishment status of the current trajectory.
Argument description	current_trajectory_state:Return the accomplishment status of the current trajectory.
Command format	{s:s,s:b}
Demo	{"state":"current_trajectory_state", "trajectory_state":true}
	Explain: The current trajectory has reached the final.

4. System Configuration

4.1 System Configuration-System Querying

(1) Query the status of the controller

Functionality	To query the status of the controller.
Argument description	get_controller_state:Query the status of the controller.
Command format	{s:s}
Demo	{"command":"get_controller_state"}
	Explain: Query the status of the controller.
Return	See Table 4.2-(1)

4.2 System Configuration-System Returning

(1) Return the status of the controller

Functionality	To return the status of the controller.
Argument description	controller_state:Return the status of the controller.
Command format	{s:s,s:i,s:i,s:i }
Demo	{"state":"controller_state","voltage":24000,"current":15000, "temperature":42000, "err_flag":0}
	Explain: Return the status of the controller. Voltage:24v,current:1.5A, temperature:42℃,error code: 0. The resolution of the voltage, current and temperature: 0.001

(2) Automatically return the system status

Functionality	To automatically return the system status.

Argument description	system_state_servo:Servo of the system status. time_cnt: Time counting in the system. Add 1 for every 20ms joint:Joint angle,resolution 0.001° pose:First 3 elements are positions, resolution: 0.001mm. The latter 3 elements are Euler angles, resolution: 0.001rad. sys_err:Error code. DI: The status of the 3 digital IO input channels at the controller end. 1-High,0-Low.
Command format	{s:s,s:b}
Demo	{"state":" system_state_servo", time_cnt:100, "joint":{1000,2000,3000,4000,5000,6000}, "pos":{1000,2000,3000, 0, 3140, 3140}, "DI":[1,1,1], "sys_err":0 }
	Explain: After starting the automatic returning, it returns the system status in 100Hz. Time counting:100 Joint angles:Joint 1—1°,Joint 2—2°,Joint 3—3°,Joint 4—4°,Joint 5—5°,Joint 6—6° Position and pose of the end:Position:x-1mm,y-2mm,z-3mm ； Euler angles:rx-0, ry-3.14, rz-3.14 System error code:0-Normal.

4.3 System Configuration-System Command

(1) Power on and off

Functionality	To power on and off the robot arm.
Argument description	set_arm_power:Power on and off the robot arm. arm_power:Power status. 1-Power on, 0-Power off.
Command format	{s:s,s:i}
Demo	{"command":"set_arm_power", "arm_power":1}
	Explain: Power on the robot arm.
Return	Format:{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"set_arm_power", "arm_power":true}

(2) Return the power status

Functionality	To return the power status of the robot arm.
Argument description	get_arm_power_state:Return the power status of the robot arm.
Command format	{s:s}
Demo	{"command":"get_arm_power_state"}
	Explain: Return the power status of the robot arm.
Return	Format:{s:s,s:i}, 1-Power on, 0-Power off.
	{"state":"arm_power_state", "power_state":1}

(3) Retrieve the software version

Functionality	To return the software version of the robot arm.
Argument	get_arm_software_version:Return the software version of the robot arm.

description	
Command format	{s:s }
Demo	{"command":"get_arm_software_version"}
	Explain: Return the software version of the robot arm.
Return	Format: {s:s,s:i,s:i,s:i,s:i}, true-Setup succeeded, false-Setup failed.
	{"state":"arm_software_version","Plan_version":7013129, "Ctrl_version":7013129, "Real-time_Kernal1":7013129, "Real-time_Kernal2":7013129} 7013129：Converted to uint32_t type of 16-feed data, 0x6B0309 6B-denotes RM65-B, 6D-denotes RM65-ZF, 6F-denotes RM65-SF 0309-denotes the software version is V3.9

(4) Return the cumulative running time of the controller

Functionality	To return the cumulative running time of the controller after manufactured.
Argument description	get_system_runtime: Return the cumulative running time of the controller.
Command format	{s:s }
Demo	{"command":"get_system_runtime"}
	Explain: Return the cumulative running time of the controller.
Return	Format:{s:s, s:i, s:i, s:i, s:i},
	{"command":"get_system_runtime", "day":0, "hour":0, "min":0, "sec":0} If the system is normal, then return the running time. {"command":"get_system_runtime", "sys_state":"sd_card_err "} If the system is abnormal, then return the error code.

(5) Clear the cumulative running time of the controller

Functionality	To clear the cumulative running time of the controller after manufactured.
Argument description	clear_system_runtime:Clear the cumulative running time of the controller.
Command format	{s:s }
Demo	{"command":"clear_system_runtime"}
	Explain: Clear the cumulative running time of the system.
Return	Format:{s:s, s:b}
	{"command":"clear_system_runtime", "clear_state":true} true-Clear succeeded, false-Clear failed.

(6) Return the cumulative rotation angle of the joint

Functionality	To return the cumulative rotation angle of the joint after manufactured.
Argument description	get_joint_odom:Return the cumulative rotation angle sof the joints.
Command format	{s:s }
Demo	{"command":"get_joint_odom"}
	Explain: Return the cumulative rotation angle of the joint.
Return	Format:{s:s, s:[i, i, i, i, i, i]}
	{"command":"get_joint_odom", "odom":[1000, 2000, 3000, 4000, 5000, 6000]}. If succeeded, return the cumulative rotation angle of the

	joint. {"command":" get_joint_odom", "sys_state":"sd_card_err"} If the system is abnormal, then return the error code.

(7) Clear the cumulative rotation angle of the joint

Functionality	To clear the cumulative rotation angle of the joint after manufactured.
Argument description	clear_joint_odom:Clear the cumulative rotation angles of the joints.
Command format	{s:s}
Demo	{"command":"clear_joint_odom"}
	Explain: Clear the cumulative rotation angle of the joint.
Return	Format:{s:s, s:b}
	{"command":"clear_joint_odom", "clear_state":true} true-Clear succeeded, false-Clear failed.

(8) Set the high-speed network port

Functionality	The controller panel has 2 network ports. The left side is the high-speed network port, which is closed by default and needs to be opened by command. The right side near the edge of the panel is the normal network port, which users do not need to configure and can use directly. Note: The IP address of the high-speed network port is 192.168.1.18 and the port number is 8080, which cannot be modified by users.
Argument description	set_high_speed_eth： To set the high-speed network port
Command format	{s:s,s:i}
Demo	{"command":"set_high_speed_eth", "mode":0}
	Explain： mode 0-Close the high-speed network port. 1- Open the high-speed network port. After successful setting, the robot arm controller buzzer will prompt, then the user can plug the network cable into the high-speed network port, restart the controller, and the user can use it after successful initialization. The configuration information will be saved in the controller and will not be lost after reboot. Note: After the controller starts, there will be an automatic query of the MAC address of PC, and it will keep waiting for the physical connection of PC and the control network port.
Return	Format： {s:s,s:b}
	{"command":"set_high_speed_eth","set_state":true} true-Setup succeeded, false- Setup failed.

4.4 System Configuration-Communication Configuration

The robot arm controller can communicate with the user through the network port, WIFI, RS232-USB and RS485 interface. The user does not need to switch when using and can use any of the above interfaces. When the controller receives the command, if the command format is correct, the data will be transmitted through the same interface.

(1) Set wifiAP

Functionality	Set wifiAP content. No returns. If the setting is successful, the buzzer rings and restart the controller to enter the WIFIAP mode.
Argument description	set_wifi_ap:Set wifiAP content.
Command format	{s:s,s:s,s:s}
Demo	{"command":"set_wifi_ap","wifi_name":"robot", "password":"12345678"}
	Explain: Set wifiAP content, wifi name:robot,pwd:12345678

(2) Set wifiSTA

Functionality	Set wifiSTA content. No returns. If the setting is successful, the buzzer rings and restart the controller to enter the WIFISTA mode.
Argument description	set_wifi_sta:Set wifiSTA content.
Command format	{s:s,s:s,s:s}
Demo	{"command":"set_wifi_sta","router_name":"robot", "password":"12345678"}
	Explain: Set wifiSTA content,wifi name:robot,pwd:12345678

(3) Set the USB communication

Functionality	Set the sampling rate of UART-USB. No returns. Communicate through USB after setting.
Argument description	set_usb:Set the baud rate of USB. Max: 460800Hz.
Command format	{s:s,s:i}
Demo	{"command":"set_usb", "baudrate":460800}
	Description: Configure USB baud rate to 460800 Baud rate optional range: 9600,38400,115200 and 460800, if the user set other values, the controller will process according to 460800 by default. The controller will record the current baud rate after the command is issued, and will still use the same baud rate for external communication after power failure and restart.

(4) Set RS485

Functionality	To configure RS485 baud rate. No returns.
Argument description	set_RS485: configure RS485 baud rate with the maximum of 60800Hz.
Command format	{s:s, s:i}
Functionality	{"command":"set_RS485","baudrate":460800}
Explanation	Description: Configure RS485 baud rate to 460800 Hz. Baud rate optional range: 9600,38400,115200 and 460800, if user set other values, the controller will process according to 460800 by default. After the command is issued, if the Modbus mode is open, it will be closed automatically. Meanwhile, the controller will record the current baud rate, and it will still use the same baud rate for external communication after

	power failure and restart.

4.5 Query the Status of the Robot Arm

(1) Query the joint angles

Functionality	To query the joint angles.
Argument description	get_joint_degree:Retrieve the joint angles.
Command format	{s:s}
Demo	{"command":"get_joint_degree "}
	Explain:Query the joint angles.
Return	See Table 4.5-(2)

(2) Return the joint angles

Functionality	To return the joint angles.
Argument description	joint_degree:The joint angles.
Command format	{s:s, s:[i,i,i,i,i,i] }
Demo	{"command":" get_joint_degree "}
	{"state":"joint_degree","joint":[10,20,30,40,50,60]} Joint angle resolution:0.001°

(3) Query the whole status of the robot arm at one time

Functionality	To query the whole status of the robot arm at one time.
Argument description	get_arm_all_state:Retrieve the whole information regarding the robot.
Command format	{s:s}
Demo	{"command":"get_arm_all_state"}
	Explain:query the whole status of the robot.
Return	See Table 4.5-(4)

(4) Return the whole status of the robot

Functionality	To Return the whole status of the robot.
Argument description	all_state:The whole status.
Command format	{s:s, s:{ s:[i,i,i,i,i,i], s:[i,i,i,i,i,i], s:[i,i,i,i,i,i], s:[i,i,i,i,i,i], s:[i,i,i,i,i,i], s:i, s:i}}
Demo	{"command":" get_arm_all_state"}
	{"state":"arm_all_state","all_state":{"temperature":[21,22,24,25,26,27], "current":[11,12,13,14,15,16],"voltage":[31,32,33,34,35,36], "err_flag":[1,2,3,4,5,6], "en_flag":[1,1,1,1,1,1],"sys_err":0}} Temperature resolution:0.001℃ Current resolution:0.001mA Voltage resolution:0.001V “err_flag”: Error code of the joint "sys_err": Error code of the robot

(5) Query the number of the trajectory

Functionality	To query the number of the trajectory.
Argument description	get_arm_plan_num:Query the number of the trajectory.
Command format	{s:s}
Demo	{"command":"get_arm_plan_num "}
	Explain:Query the number of the trajectory.
Return	See Table 4.5-(6)

(6) Return the number of the trajectory

Functionality	To return the number of the trajectory.
Argument description	plan_num:Return the number of the trajectory.
Command format	{s:s, s:i }
Demo	{"command":" get_arm_plan_num "}
	{"state":"arm_plan_num", "plan_num":1:} Run the first trajectory.

4.6 Configuration and Retrieval of the Controller IOs

The number and types of the controller IOs are as follows:

Digital output:DO	4 channels,configurable 0~12V
Digital input:DI	3 channels,configurable 0~12V
Analog output:AO	4 channels,configurable 0~10V
Analog input:AI	4 channels,configurable 0~10V

(1) Set the digital output

Functionality	To set the digital output.
Argument description	set_DO_state:Set the digital output.
Command format	{s:s, s:i, s:i}
Demo	{"command":"set_DO_state","IO_Num":1, "state":1}
	Explain:”IO_Num”:IO port #,range:1~4 “state”:IO status,1-Output high,0-Output low.
Return	{"command":"set_DO_state", "state":true} Setup succeeded. {"command":"set_DO_state", "state":false} Setup failed.

(2) Retrieve the status of the digital output

Functionality	To retrieve the status of the digital output.
Argument description	get_DO_state:Retrieve the status of the digital output.
Command format	{s:s, s:i }
Demo	{"command":"get_DO_state","IO_Num":1 }
	Explain:”IO_Num”:IO port #,range:1~4
Return	{"state":"DO_state","IO_Num":1, "IO_state":1}

	“state”:IO status,1-Output high,0-Output low.

(3) Retrieve the status of the digital input

Functionality	To retrieve the status of the digital input.
Argument description	get_DI_state:Retrieve the status of the digital input.
Command format	{s:s, s:i }
Demo	{"command":"get_DI_state","IO_Num":1}
	Explain:”IO_Num”:IO port #,range:1~3
Return	{"state":"DI_state",”IO_Num”:1, “IO_state”:1} “state”:IO status,1-Input high,0-Input low.

(4) Set the analog output

Functionality	To set the analog output.
Argument description	set_AO_state:Set the analog output.
Command format	{s:s, s:i, s:i}
Demo	{"command":"set_AO_state","IO_Num":1, "voltage":1000}
	Explain:”IO_Num”:IO port #,range:1~4 “voltage”:IO output voltage,resolution 0.001V,range:0~10000,representative output voltage 0v~10v
Return	{"command":"set_AO_state", "state":true} Setup succeeded {"command":"set_AO_state", "state":false} Setup failed

(5) Retrieve the status of analog output

Functionality	To retrieve the status of analog output.
Argument description	get_AO_state:Retrieve the status of analog output.
Command format	{s:s, s:i}
Demo	{"command":"get_AO_state","IO_Num":1}
	Explain:”IO_Num”:IO port #,range:1~4
Return	{"state":"AO_state","IO_Num":1, “voltage”:1000} “voltage”:IO output voltage,resolution 0.001V,range:0~10000,representative output voltage 0v~10v

(6) Retrieve the status of analog input

Functionality	To retrieve the status of analog input.
Argument description	get_AI_state:Retrieve the status of analog input.
Command format	{s:s, s:i, s:i}
Demo	{"command":"get_AI_state","IO_Num":1 }
	Explain:”IO_Num”:IO port #,range:1~4
Return	{"state":"AI_state","IO_Num":1, “voltage”:1000} “voltage”:IO input voltage,resolution 0.001V,range:0~10000,representative input voltage 0v~10v

(7) Retrieve the status of all inputs

Functionality	To retrieve the status of all inputs.

Argument description	get_IO_input:Retrieve the status of all inputs.
Command format	{s:s }
Demo	{"command":"get_IO_input"}
Return	{"state":"IO_input_state","DI":[1,1,1], "AI":[1000, 2000, 3000, 4000]} ”DI”: Digital input status,1-High,0-Low “AI”: Analog input voltage,resolution 0.001V,i.e. 1000 denotes 1V.

(8) Retrieve the status of all outputs.

Functionality	To retrieve the status of all outputs.
Argument description	get_IO_output:Retrieve the status of all outputs.
Command format	{s:s }
Demo	{"command":"get_IO_output" }
Return	{"state":"IO_output_state","DO":[1,1,1,1],"AO":[1000, 2000, 3000]} ”DI”: Digital output status,1-High,0-Low “AI”: Analog output voltage,resolution 0.001V,i.e. 1000 denotes 1V.

4.7 The Arm-End IOs

The number and types of the arm-end IOs are as follows:

Power output	1 Channel,configurable of 0V/5V/12V/24V
Digital IO	2 Channels, and I/O can be configurable. Input：reference voltage 12V~24V Output：5~24V, the same as the power output.
Communication interface	1 Channel,configurable RS485

(1) Set the digital output of the arm end

Functionality	To set the digital output of the arm end.
Argument description	set_tool_DO_state:Set the digital output of the arm end.
Command format	{s:s, s:i, s:i}
Demo	{"command":"set_tool_DO_state","IO_Num":1, "state":1}
	Explain:”IO_Num”:IO port #,range:1~2 “state”:IO status,1-Output high,0-Output low.
Return	{"command":"set_tool_DO_state", "set_state":true} Setup succeeded {"command":"set_tool_DO_state", "set_state":false} Setup failed
Comment	Digital IO is input and output reusable. When the output command is issued, the end digital IO will automatically change to output mode.

(2) Set the tool end IO mode

Functionality	To set digital IO mode
Argument description	set_tool_IO_mode：Set digital IO mode
Command format	{s:s,s:i,s:i}
Demo	1. {"command":"set_tool_IO_mode","IO_Num":1,"state":0}
	Explain：”IO_Num”：IO port number with range of 1~2

	"state"：Mode，0-input，1-output.
Return	{"command":"set_tool_IO_mode","set_state":true} Setup succeeded {"command":"set_tool_IO_mode","set_state":false} Setup failed

(3) Retrieve the status of the tool end digital IO

Functionality	To retrieve the status of the digital IO.
Argument description	get_tool_IO_state: Retrieve the status of the digital IO.
Command format	{s:s, s:i }
Demo	{"command":"get_tool_IO_state"}
Return	{"state":"tool_IO_state","IO_Mode":[0,1],"IO_State":[0,1]} Explain："IO_Mode"：0-input，1-output "IO_State"：0-low，1-high.

(4) Set the power output of the tool end

Functionality	To set the power output.
Argument description	set_tool_voltage:Set the power output.
Command format	{s:s, s:i }
Demo	{"command":"set_tool_voltage", "voltage_type":1}
Explain:	“voltage_type”:Power output type,range:0~3 0-0V, 1-5V, 2-12V, 3-24V
Return	{"command":" set_tool_voltage ", "state":true} Setup succeeded {"command":" set_tool_voltage ", "state":false} Setup failed

(5) Retrieve the power output of the tool end

Functionality	To retrieve the power output type.
Argument description	get_tool_voltage:Retrieve the power output of the tool end.
Command format	{s:s }
Demo	{"command":"get_tool_voltage"}
Return	{"state":"tool_voltage_state", “voltage_type”:1} “voltage_type”:Power output type,range:0~3 0-0V, 1-5V, 2-12V, 3-24V

4.8 Arm End Tool—Gripper (Optional accessory)

The end of RM-65 robot arm is equipped with EG2-4C2 gripper from Inspire Robots Company. To facilitate the user to operate the gripper, the robot arm controller opens the control protocol of the gripper to the user, as shown below.

(1) Set the gripper route

Functionality	To set the gripper route, i.e., the maximum and minimum values of the opening of the gripper. It will be saved automatically after successful setting. It will not be lost in the power failure of the gripper.
Argument description	set_gripper_route:Set the gripper route.

Command format	{s:s, s:i, s:i}
Demo	{"command":"set_gripper_route", "min":70, "max":500 }
Explain	“min”: The minimum value of gripper’s opening,range:0~1000,not unit. “max”: The maximum value of gripper’s opening,range:0~1000,not unit.
Return	{"command":"set_gripper_route", "state":true} Setup succeeded {"command":" set_gripper_route", "state":false} Setup failed

(2) Release the gripper

Functionality	To release the gripper, i.e., the gripper moves at the specified speed to the maximum opening.
Argument description	set_gripper_release:Release the gripper.
Command format	{s:s, s:i }
Demo	{"command":"set_gripper_release", "speed":500 }
Explain	“speed”: The releasing speed,range 1~1000,no unit.
Return	{"command":" set_gripper", "state":true} Release succeeded. {"command":" set_gripper", "state":false} Release failed.

(3) Clamp at a certain speed and force

Functionality	To clamp at a certain speed and force. The gripper is clamped at the set speed and force. When the clamping force exceeds the set force threshold, the clamping is stopped.
Argument description	set_gripper_pick:Set the gripper clamping.
Command format	{s:s, s:i, s:i }
Demo	{"command":"set_gripper_pick", "speed":500, "force":200}
Explain	“speed”: Clamping speed,range 1~1000,no unit. “force”: Clamping force threshold,range 50~1000,no unit.
Return	{"command":"set_gripper", "state":true} Clamping succeeded. {"command":"set_gripper", "state":false} Clamping failed.

(4) Clamp at a certain speed and a continuous force

Functionality	To clamp at a certain speed and force. The gripper is clamped at the set speed and force. When the clamping force exceeds the set force threshold, the clamping is stopped. When the clamping force is less than the torque threshold again, the gripper is clamped again until the clamping force exceeds the force control threshold.
Argument description	set_gripper_pick_on:Set the gripper clamping.
Command format	{s:s, s:i, s:i }
Demo	{"command":"set_gripper_pick_on", "speed":500, "force":200}
Explain	“speed”: Clamping speed,range 1~1000,no unit. “force”: Clamping force threshold,range 50~1000,no unit.
Return	{"command":"set_gripper", "state":true} Clamping succeeded. {"command":"set_gripper", "state":false} Clamping failed.

(5) Set a specified position to reach

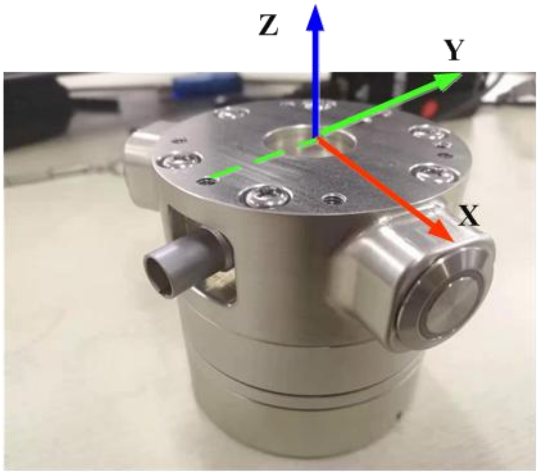
Functionality	When the current opening is smaller than the specified opening, the gripper will be released to the specified opening at the specified speed. When the current opening is larger than the specified opening, the gripper will close at the specified opening with the specified speed and torque. When the clamping force exceeds the torque threshold or reaches the specified position, the gripper will stop.
Argument description	set_gripper_position:Set a specified position for the gripper to reach.
Command format	{s:s, s:i}
Demo	{"command":"set_gripper_position", "position":500}
Explain	"position":The opening position of the gripper,range:1~1000,no unit.
Return	{"command":"set_gripper", "state":true} Reach the position successfully. {"command":"set_gripper", "state":false} Fail to reach.

4.9 Arm-End Tool—Six-Axis Force (Optional)

RM RM-65F robot arm end is equipped with an integrated six-axis force sensor, which requires no external wiring. Users can operate the six-axis force directly through the protocol and obtain the six-axis force data.

As shown in the figure below, the Z axis of the six-axis force is directing upward, and the Y-axis of the six-axis force is in the opposite direction of the navigation interpolation. The coordinate system applies to the right hand rule. When the robot arm is in the origin position, the direction of the base coordinate system is the same as that of the six-axis force coordinate system.

In addition, the six-axis force rated force is 200N, rated torque is 8Nm, overload level is 300%FS, working temperature is 5~80℃, and accuracy is0.5%FS. Pay attention to the instructions during the usage to prevent damage to the six-axis force sensor.



(1) Query the data of the six-axis force sensor

Functionality	To query the force and torque of the six-axis force sensor:Fx, Fy, Fz, Mx,

	My, Mz
Argument description	get_force_data: Query force sensor information. If force data is to be obtained periodically, the period should not be less than 50ms.
Command format	{s:s }
Demo	{"command":"get_force_data"}
Explanation	N/A
Return	{"command":"get_force_data", "force_data":{1000, 2000, 3000, 400, 500, 600} } Resolution:0.001 force_data: Fx=1N, Fy=2N, Fz=3N, Mx=0.4Nm,My=0.5Nm, Mz=0.6Nm.

(2) Clear the data of the six-axis force sensor

Functionality	To clear the data of the six-axis force sensor.
Argument description	clear_force_data:Data of the six-axis force sensor.
Command format	{s:s}
Demo	{"command":"clear_force_data" }
Explanation	N/A
Return	{"command":"clear_force_data", "clear_state":true} Clear succeeded. {"command":"clear_force_data", "clear_state":false} Clear failed.

(3) Automatically set the center of mass of the six-axis force sensor

Functionality	To set the center of mass of the six-axis force. After the six-axis force is reinstalled, the initial force and center of mass received by the six-axis force must be recalculated. The data of the six-axis forces under different poses are obtained and used to calculate the position of the center of gravity. After the instruction is issued, the robot arm moves to each calibration point at a certain speed. The process cannot be interrupted. After the interruption, it must be recalibrated.
	
	Note: It is necessary to ensure that the robot arm is calibrated in stationary state.
	Joint angles of Point 1:{0,0,0,0,0,0}
	Joint angles of Point 2:{0,0,90,0,90,0}
	Joint angles of Point 3:{0,0,0,0,90,0}
	Joint angles of Point 4:{0,0,0,0,90,-90}

Argument description	set_force_sensor: Set the value when the force sensor is at the specified position.
Command format	{s:s }
Demo	{"command":"set_force_sensor "}
Explanation	N/A
Return	{"command":" set_force_sensor ", "set_state":true} Setup succeeded {"command":" set_force_sensor ", "set_state":false} Setup failed

(4) Manually set the center of mass of the six-axis force sensor

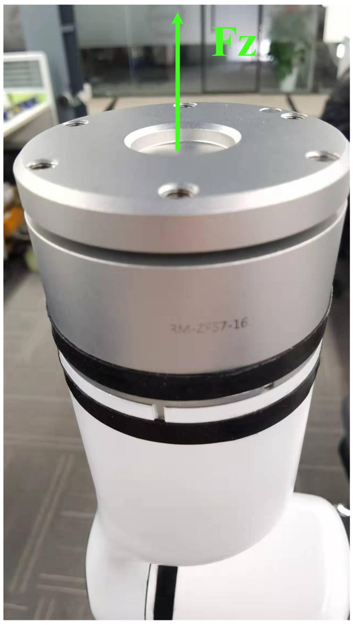
Functionality	To set the center of gravity of the six-axis force sensor. After six-axis force sensor is installed, it must recalculate the initial force and center of gravity of the six-axis force sensor. This manual calibration process, which is suitable for limited working space area to prevent the robot arm from collision during automatic calibration. The user needs to manually select four poses, and when the four poses are sent, the robot arm starts to automatically move along the target poses set by the user and calculates the center of gravity of the six-axis force sensor in the process.
Argument description	manual_set_force：Calibrate the center of gravity/mass manually. pose1: Joint angle at position 1; pose2: Joint angle at position 2; pose3: Joint angle at position 3; pose4: Joint angle at position 4; The above 4 positions must be sent to the robot in order. When the pose4 is sent, the robot arm starts to run automatically to calculate the center of gravity, and returns to the protocol after the calculation is completed.
Command format	{s:s,s:[i,i,i,i,i,i] }
Demo	{"command":"manual_set_force_pose1", "joint":[0, 0, 0, 0, 90000, 0]}
Explanation	joint： precision of 0.001°, namely the target angles of Joint 1~6 at the position 1 are 0°， 0°， 0°， 0°， 90°， and 0°, respectively.
Return	{"command":"set_force_sensor","set_state":true} Setup succeeded {"command":"set_force_sensor","set_state":false} Setup failed

(5) Stop calibrate the center of mass of the six-axis force sensor

Functionality	If there is an accident in the process of calibration, the command will be sent to stop the movement of the robot and exit the calibration process.
Argument description	stop_set_force_sensor: Stop calculating the center of gravity position of the force sensor.
Command format	{s:s }
Demo	{"command":"stop_set_force_sensor "}
Explanation	
Return	{"command":" stop_set_force_sensor ", "stop_state":true} Calculation succeeded. {"command":" stop_set_force_sensor ", "stop_state":false} Calculation failed.

4.10 Arm-End Tool—One-Axis Force Sensor (Optional)

RM robot arm end interface board integrates with a one-axis force sensor to obtain the force in the Z-direction, range 200N, accuracy 0.5%FS.



(1) Query the data of the one-axis force sensor

Functionality	To query the one-axis force sensor's data.
Argument description	get_Fz：Query the force sensor data.
Command format	{s:s}
Demo	{"command":"get_Fz"}
Explanation	Note: After the first frame of command is issued, it starts to update the one-axis force data therefore the return data has time lag. So please start using the data from the second frame. If Fz data is queried periodically, the frequency cannot be higher than 40Hz.
Return	{"command":"get_Fz","Fz":12000} If succeeded, return the data. Precision：0.001N, e.g., Fz in this demo is 12N. {"command":"get_Fz","set_state":false} If failed, return the command.

(2) Clear the data of the one-axis force sensor

Functionality	To clear the data of the one-axis force sensor.
Argument description	clear_Fz: After clearing the one-axis force data, all subsequent acquired data are based on the current bias.
Command format	{s:s}
Demo	{"command":"clear_Fz"}
Explanation	N/A
Return	{"command":"clear_Fz","set_state":true} Setup succeeded. {"command":"clear_Fz","set_state":false}Setup failed.

(3) Automatically set the center of mass of the one-axis force sensor

Functionality	To set the parameters of the center of gravity of the one-axis force sensor. After the one-axis force sensor is reinstalled, the initial force and the center of gravity to which the one-axis force sensor is subjected must be recalculated. Obtain the data of the one-axis force sensor in different poses

	to calculate the center of gravity position. This step is important for one-axis force-based force-position hybrid control operation.
Argument description	auto_set_Fz：Calibration sensor’s origin data.
Command format	{s:s}
Demo	{"command":" auto_set_Fz "}
Explanation	
Return	{"command":"set_force_sensor","set_state":true}Calibration succeeded. {"command":"set_force_sensor","set_state":false}Calibration failed.

(4) Manually set the center of mass of the one-axis force sensor

Functionality	To set the center of gravity of the one-axis force sensor. After one-axis force sensor is installed, it must recalculate the initial force and center of gravity of the six-axis force sensor. This manual calibration process, which is suitable for limited working space area to prevent the robot arm from collision during automatic calibration. The user needs to manually select 2 poses, and when the four poses are sent, the robot arm starts to automatically move along the target poses set by the user and calculates the center of gravity of the one-axis force sensor in the process.
Argument description	manual_set_Fz：Calibrate the origin of the sensor pose1: joint angle at position 1; pose2:joint angle at position 2;
Command format	{s:s,s:[i,i,i,i,i,i] ,s:[i,i,i,i,i,i] }
Demo	{"command":"manual_set_Fz", "pose1":[0, 0, 0, 0, 0, 0], "pose2":[0, 0, 90000, 0, 90000, 0]}
Explanation	pose1：joint angle at position 1 with precision of 0.001°. That is, 90000 represents 90°.
Return	{"command":"set_force_sensor","set_state":true}Calibration succeeded. {"command":"set_force_sensor","set_state":false} Calibration failed.

(5) Stop calibrate the center of mass of the one-axis force sensor

Functionality	If there is an accident in the process of calibration, the command will be sent to stop the movement of the robot and exit the calibration process.
Argument description	stop_set_force_sensor：Stop calculating the center of gravity position of the force sensor.
Command format	{s:s}
Demo	{"command":"stop_set_force_sensor"}
Explanation	
Return	{"command":"stop_set_force_sensor","stop_state":true} Calculation succeeded. {"command":"stop_set_force_sensor","stop_state":false} Calculation failed.

4.11 Drag Teaching -Trajectory Recurrence

In the drag teaching process, the track points can be recorded, and the trajectory can be reproduced according to the user's instructions.

(1) Start the drag teaching

Functionality	To start the drag teaching.
Argument description	start_drag_teach:Start the drag teaching. trajectory_record: Record the trajectory by drag teaching. 0-don't record,1-record.
Command format	{s:s, s:i }
Demo	{"command":"start_drag_teach", "trajectory_record":1}
	Explain: Start the drag teaching and record the trajectory.
Return	Format:{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"start_drag_teach","drag_teach":true}

(2) Stop the drag teaching

Functionality	To stop the drag teaching.
Argument description	stop_drag_teach:Stop the drag teaching.
Command format	{s:s }
Demo	{"command":"stop_drag_teach"}
	Explain: Stop the drag teaching.
Return	Format:{s:s,s:b}, true-Setup succeeded, false-Setup failed.
	{"command":"stop_drag_teach","drag_teach":true}

(3) Start the composite drag teaching mode

Functionality	To start the composite drag teaching mode.
Argument description	start_multi_drag_teach: Start the composite drag teaching.
Command format	{s:s, s:i}
Demo	{"command":"start_multi_drag_teach", "mode":0}
	Explain: mode: drag teaching mode. 0-current loop mode, 1-use end six-axis force, move position only, 2- use end six-axis force, move pose only, 3- use end six-axis force, move both position and pose.
Return	{"command":"start_multi_drag_teach", "set_state":true} Setup succeeded {"command":"start_multi_drag_teach", "set_state":false} Setup failed.

(4) Start the trajectory recurrence

Functionality	To reproduce the trajectory recorded by dragging. Note:It must be used after the end of dragging, and ensure that the robot arm is located at the starting point of dragging instruction.
Argument description	run_drag_trajectory:Start to reproduce the recorded trajectory.
Command format	{s:s }
Demo	{"command":"run_drag_trajectory "}

Explanation	
Return	{"command":" run_drag_trajectory ", " run_state":true} Recurrence succeeded. {"command":" run_drag_trajectory", "run_state":false} Recurrence failed.

(5) Pause the trajectory recurrence

Functionality	To pause the trajectory recurrence.
Argument description	pause_drag_trajectory:Pause the trajectory recurrence process.
Command format	{s:s }
Demo	{"command":"pause_drag_trajectory "}
Explanation	
Return	{"command":" pause_drag_trajectory ", " pause_state":true} Pause succeeded. {"command":"pasuse_drag_trajectory", "pause_state":false} Pause failed.

(6) Continue the trajectory recurrence

Functionality	To continue the paused trajectory recurrence. Note:When the paused trajectory continues, it must ensure that the robot arm is at the position when it was paused, otherwise an error will be reported, and the user can only reproduce the trajectory from the beginning position.
Argument description	continue_drag_trajectory:Continue the paused trajectory recurrence.
Command format	{s:s }
Demo	{"command":"continue _drag_trajectory "}
Explanation	
Return	{"command":" continue_drag_trajectory ", "continue_state":true} Continue succeeded. {"command":"continue_drag_trajectory", "continue_state":false} Continue failed.

(7) Stop the trajectory recurrence

Functionality	To stop the trajectory recurrence.
Argument description	stop_drag_trajectory:Stop the trajectory recurrence.
Command format	{s:s }
Demo	{"command":"stop _drag_trajectory "}
Explanation	
Return	{"command":"stop_drag_trajectory ", "stop_state":true} Stop succeeded. {"command":"stop_drag_trajectory", "stop_state":false} Stop failed.

(8) Move to the trajectory starting point

Functionality	Before the trajectory is reproduced, the robot arm must be at the starting

	point of the trajectory. If set correctly, the robot arm will move to the starting point of the trajectory at a speed of 20%.
Argument description	drag_trajectory_origin:The starting point of the trajectory.
Command format	{s:s }
Demo	{"command":"drag_trajectory_origin"}
Explanation	
Return	{"state":"current_trajectory_state", "trajectory_state":true} Movement succeeded. {"state":"current_trajectory_state", "trajectory_state":false } Movement failed.

(9) Set the hybrid force-position control

Functionality	In trajectory planning in Cartesian space, the contact force at the end of the robot arm can be kept constant by using this feature.
Argument description	set_force_position: Set the hybrid force-position control.
Command format	{s:s, s:i, s:i}
Demo	{"command":"set_force_position", "mode":1, "force_level":1}
Explanation	Mode:1-Base coordinate system, Z axis: force control；2- force control of operation surface；3-radial force control of operation surface. force_level: Force control level, and the higher the level, the smaller the force.
Return	{"state":" set_force_position ", "set_state":true} Setup succeeded. {"state":" set_force_position ", "set_state":false} Setup failed.

(10) Stop the hybrid force-position control

Functionality	To stop the hybrid force-position control mode.
Argument description	stop_force_position: Stop the hybrid force-position control mode.
Command format	{s:s }
Demo	{"command":"stop_force_position "}
Explain	
Return	{"state":" stop_force_position ", "stop_state":true} Stop succeeded. {"state":" stop_force_position ", "stop_state":false} Stop failed.

4.12 Dexterous Five-Fingers Hand (optional)

RM robot arm end can install the dexterous hand tool by Inspire Robots Company. The dexterous hand can be set by protocol.

(1) Set the dexterous hand gesture

Functionality	To set the dexterous hand gesture.
Argument description	set_hand_posture:Set the dexterous hand gestures. posture_num: Pre-saved gesture number in dexterous hand,range:1~40
Command format	{s:s, s:i}
Demo	{"command":"set_hand_posture", "posture_num":1}

Explanation	Set the dexterous hand to execute Gesture 1.
Return	{" command ":"set_hand_posture", "set_state":true} Setup succeeded. {" command ":"set_hand_posture", "set_state":false}Setup failed.

(2) Set the dexterous hand movement order

Functionality	To Set the dexterous hand movement order.
Argument description	set_hand_seq:Set the dexterous hand gestures. seq_num: Pre-saved gesture number in dexterous hand,range:1~40
Command format	{s:s, s:i}
Demo	{"command":"set_hand_seq", "seq_num":1}
Explanation	Set the dexterous hand to execute Gesture 1.
Return	{" command ":"set_hand_seq", "set_state":true} Setup succeeded. {" command ":"set_hand_seq", "set_state":false}Setup failed.

(3) Set the angles of each DoF at the dexterous hand

Functionality	To set the angles of dexterous hand. Dexterous hand has 6 degrees of freedom from 1 to 6. They are the little finger, ring finger, middle finger, index finger, thumb bending, and thumb rotation, respectively.
Argument description	set_hand_angle:Set the angles of dexterous hand. hand_angle: An array of finger angles, ranging from 0 to 1000. In addition, -1 means that the degree of freedom does not perform any action and remains in the current state.
Command format	{s:s, s:[i,i,i,i,i,i]}
Demo	{"command":"set_hand_angle ", "hand_angle":[-1, 100, 200, 300, 400, 500]}
Explanation	Set each finger movement of dexterous hand
Return	{" command ":"set_hand_angle", "set_state":true} Setup succeeded. {" command ":"set_hand_angle", "set_state":false}Setup failed.

(4) Set the dexterous hand speed

Functionality	To set the dexterous hand speed.
Argument description	set_hand_speed:Set the dexterous hand speed. hand_speed: The dexterous hand speed,range:1~1000
Command format	{s:s, s:i }
Demo	{"command":"set_hand_speed ", "hand_speed":500}
Explanation	Set the dexterous hand speed.
Return	{" command ":"set_hand_speed", "set_state":true}Setup succeeded. {" command ":"set_hand_speed", "set_state":false}Setup failed.

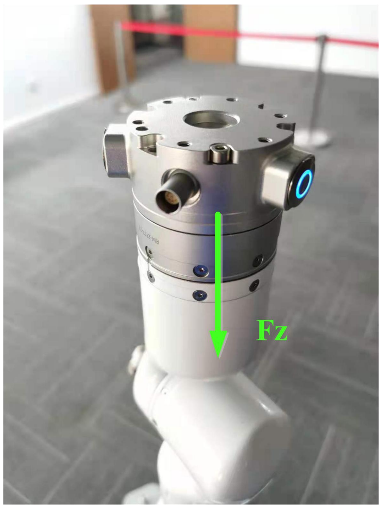
(5) Set the force threshold of the dexterous hand

Functionality	To set the force threshold of the dexterous hand.
Argument description	set_hand_force:Set the force threshold of the dexterous hand. hand_force: The dexterous hand force,range:1~1000
Command format	{s:s, s:i }
Demo	{"command":"set_hand_force", "hand_force":500}
Explanation	Set the force threshold of the dexterous hand.

Return	{" command ":"set_hand_force", "set_state":true}Setup succeeded. {" command ":"set_hand_force", "set_state":false}Setup failed.

4.13 Arm-End Sensor: One-Axis Force Sensor (optional)

The interface plate at RM arm end is integrated with a one-axis force sensor, which can obtain the force in the Z direction. The measuring range is 200N, and the accuracy is 0.5%FS.



(1) Query the arm-end one-axis force data

Functionality	To query the arm-end one-axis force data.
Argument description	get_Fz: Query the arm-end one-axis force data.
Command format	{s:s }
Demo	{"command":"get_Fz"}
Explanation	Note: After the first command is issued, the one-axis force data is updated, and the returned first data frame at this time has a lag. Please start with the valid data from the second data frame. If Fz data is queried periodically, the frequency cannot be higher than 40Hz.
Return	{" command ":"get_Fz", "Fz":12000} If succeeded, return data. With resolution: 0.001N, this demo Fz is 12N. {" command ":"get_Fz", "set_state":false} If failed, return the command.

(2) Clear the arm-end one-axis force data

Functionality	To clear the arm-end one-axis force data.
Argument description	clear_Fz: After the one-axis force data is cleared, all subsequent data obtained are based on the current bias.
Command format	{s:s}
Demo	{"command":"clear_Fz"}
Explanation	N/A

Return	{" command ":"clear_Fz", "set_state":true} Setup succeeded. {" command ":"clear_Fz", "set_state":false} Setup failed.

4.14 Modbus RTU Configuration

RM robot arm has one RS485 communication each at the controller's 26-core and end interface board 9-core aviation plugs, both of which can be configured in standard ModbusRTU mode via the JSON protocol. The peripherals of the port connection are then read and written through the JOSN protocol.

Note: The controller's RS485 interface can be used for user control of the robot arm when it is not configured as ModbusRTU mode, and these two modes (ModbusRTU and non ModbusRTU) are not compatible. To use the robot arm control mode, the ModbusRTU mode must be turned off. After the Modbus RTU mode is turned off, the system will automatically switch back to robot arm control mode with baud rate of 460800BPS, stop bit of 1, data bit of 8, and no check.

(1) Set Modbus RTU mode

Functionality	To set the communication in ModbusRTU mode. After the robot arm starts, this step must be conducted first before do anything via the communication port, or an error message will be returned. In addition, the robot arm will save the user's configuration, and the arm will automatically revert to the mode configured before when restart.
Argument description	set_modbus_mode：Set ModbusRTU mode
Command format	{s:s, s:i, s:i, s:i}
Demo	{"command":"set_modbus_mode", "port":0, "baudrate":115200, "timeout":1}
Explanation	port: communication port， 0-controller RS485, 1-end interface board RS485 baudrate：baud rate，supporting 9600,115200,and 460800 baud rates. timeout: timeout in seconds. All read and write instructions for the Modbus device that do not return response data within the specified timeout time will return a timeout error. The timeout time cannot be 0, and if set to 0, the robot arm is configured by 1. Other configurations default values: data bit-8, stop bit-1, parity-none.
Return	{"command":"set_modbus_mode", "set_state":true} Setup succeeded {"command":"set_modbus_mode", "set_state":false} Setup failed.

(2) Close ModbusRTU mode

Functionality	To close ModbusRTU mode.
Argument description	close_modbus_mode：close ModbusRTU mode
Command format	{s:s}
Demo	{"command":"close_modbus_mode", "port":0}
Explanation	When it is closed, the port will not respond to any read or write instructions. port:port number，0-controller RS485, 1-end interface board RS485

Return	{"command":"close_modbus_mode", "set_state":true} Setup succeeded {"command":"close_modbus_mode", "set_state":false} Setup failed.

(3) Read coil

Functionality	To read coil(s).
Argument description	read_coils：read coil(s)
Command format	{s:s, s:i, s:i, s:i, s:i}
Demo	{"command":"read_coils", "port":0, "address":10, "num":2, "device":2}
Explanation	port:port number，0-controller RS485, 1-end interface board RS485 address：coil start address num：the number of coils to be read. This command supports reading up to 8 coil data at a time, i.e., the data returned will not be one byte. device：peripheral device address
Return	{"state":"coils_data", "data":8} Read successfully, return coil state data type：int8 {"state":"coils_data", "read_state":false} Write failed;No data was obtained before the timeout.

(4) Read discrete input

Functionality	To read discrete input.
Argument description	read_input_status：read discrete input.
Command format	{s:s, s:i, s:i, s:i, s:i}
Demo	{"command":"read_input_status", "port":0, "address":10, "num":2, "device":2}
Explanation	port:port number，0-controller RS485, 1-end interface board RS485 address：data start address num：the number of coils to be read. This command supports reading up to 8 coil data at a time, i.e., the data returned will not be one byte. device：peripheral device address
Return	{"state":"input_status", "data":8} Read successfully, return discrete coil state data type：int8 {"state":"input_status", "read_state":false} Write failed;No data was obtained before the timeout.

(5) Read holding register

Functionality	To read holding register.
Argument description	read_holding_registers：read holding registers.
Command format	{s:s, s:i, s:i, s:i }
Demo	{"command":"read_holding_registers", "port":0, "address":10, "device":2}
Explanation	port:port number，0-controller RS485, 1-end interface board RS485

	address：data start address. This command can only read 1 register at a time, i.e., 2 bytes of data, and cannot read more than one register data at a time. device：peripheral device address
Return	{"state":"holding_registers", "data":8}　　If read successfully, return register data. data type：int16 {"state":"holding_registers", "read_state":false}　Write failed;No data was obtained before the timeout.

(6) Read input register

Functionality	To read input register
Argument description	read_input_registers：read input register.
Command format	{s:s, s:i, s:i, s:i}
Demo	{"command":"read_input_registers", "port":0, "address":10, "device":2}
Explanation	port:port number，0-controller RS485, 1-end interface board RS485 address：data start address. This command can only read 1 register at a time, i.e., 2 bytes of data, and cannot read more than one register data at a time. device：peripheral device address
Return	{"state":"input_registers", "data":8}　　If read successfully, return register data. data type：int16 {"state":"input_registers", "read_state":false}　Write failed;No data was obtained before the timeout.

(7) Write single coil

Functionality	To write single coil data.
Argument description	write_single_coil：Write single coil data.
Command format	{s:s, s:i, s:i, s:i, s:i}
Demo	{"command":"write_single_coil", "port":0, "address":10, "data":1000, "device":2}
Explanation	port:port number，0-controller RS485, 1-end interface board RS485 address：coil start address data：coil data to be written data type：int16 device：peripheral device address
Return	{"command":"write_single_coil", "write_state":true} Write succeeded. {"command":"write_single_coil", "write_state":false}　Write failed;No data was obtained before the timeout, or the command content was incorrect.

(8) Write single register

Functionality	To write single register
Argument	write_single_register：Write single register

description	
Command format	{s:s, s:i, s:i, s:i, s:i}
Demo	1. {"command":"write_single_register", "port":0, "address":10, "data":1000, "device":2}
Explanation	port:port number， 0-controller RS485, 1-end interface board RS485 address： register start address data： register data to be written， data type： int16 device： peripheral device address
Return	{"command":"write_single_register", "write_state":true} Write succeeded. {"command":"write_single_register", "write_state":false} Write failed;No data was obtained before the timeout, or the command content was incorrect.

4.15 System Installation and Joint Version Information

RM robot arm supports different forms of installation, but the installation methods vary, as do the robot's dynamic model parameters and coordinate system orientation.

(1) Set the installation orientation parameters

Functionality	To set the installation orientation parameters.
Argument description	set_install_pose: Set the robot base installation method.
Command format	{s:s, s:[i,i]}
Demo	{"command":"set_install_pose", "pose":[0,3141]}
Explanation	Pose: The rotation angle and pitch angle of the base relative to the horizontal surface, accuracy: 0.001 radian. As shown above, that is, the rotation angle is 0, the pitch angle is 3.141 radian.
Return	{"command":"set_install_pose", "set_state":true} Setup succeeded {"command":"set_install_pose", "set_state":false} Setup failed.

(2) Query the joint software version

Functionality	To query the joint software version.
Argument description	get_joint_software_version: Query the joint software version.
Command format	{s:s}
Demo	{"command":"get_joint_software_version"}
Explanation	N/A
Return	{s:s,s:[i,i,i,i,i,i]}
	{"state":"joint_software_version", "version":[531,531,531,531,531,531]}
	531 is of uint16， as of hexadecimal：0x0213， i.e., the version number of the current joint is 2.13

(3) Query the end interface board software version

Functionality	To query the end interface board software version.

Argument description	get_tool_software_version: Query the end interface board software version.
Command format	{s:s}
Demo	{"command":"get_tool_software_version"}
Explanation	N/A
Return	{s:s,s:i }
	{"state":"tool_software_version", "version":531}
	531 is of uint16，as of hexadecimal：0x0213，i.e., the version number of the current joint is 2.13

4.16 Passthrough Force-Position Hybrid Control Compensation (Optional)

For RM robot arms with one-axis force sensor and six-axis force sensor, the user can not only call the underlying force-position hybrid control module directly using the teach pendant software, but can also compensate for custom trajectories in the form of periodic passthrough in combination with the underlying force-position hybrid control algorithm.

(1) Start the passthrough force-position hybrid control mode

Functionality	To start the passthrough force-position hybrid control mode. This command must be issued to enable the function before passthrough.
Argument description	Start_Force_Position_Move
Command format	{s:s}
Demo	{"command":"Start_Force_Position_Move"}
Explanation	N/A
Return	{s:s,s:b}
	{"command":"Start_Force_Position_Move","set_state":true}
	True： Setup succeeded and can perform subsequent passthrough. False： Setup failed. There is error and cannot perform passthrough.

(2)Force_Position_Move： passthrough force-position hybrid compensation

Functionality	Force_Position_Move： The user periodically sends the target angle or target pose, and the force-position compensation is achieved through the one-axis force sensor or the six-axis force sensor using the robot arm's underlying force- position hybrid control module. Note 1: This function is only applicable to the robot version with the one-axis force sensor or the six-axis force sensor. Note 2: The faster the transmission/passthrough period, the better the force-position hybrid control effect; the transmission period is as fast as 20ms for WIFI and network port mode, 10ms for USB and RS485 mode; the transmission period is as fast as 10ms for high-speed network port, but

	you need to open the configuration with the command before using the high-speed network port.
Argument description	Force_Position_Move：passthrough force-position hybrid compensation pose: target pose in current coordinate system, position accuracy: 0.001mm, pose accuracy: 0.001rad joint: the target joint angle, accuracy 0.001° sensor: type of sensor used, 0-one-axis force, 1-six-axis force mode: mode, 0 - along the base coordinate system, 1 - along the tool end coordinate system dir: force control direction, 0~5 represent X/Y/Z/Rx/Ry/Rz respectively, where the default direction is Z direction for one-axis force type force: force size, precision 0.1N or 0.1Nm
Command format	{s:s,s:[i,i,i,i,i,i],s:i,s:i,s:i,s:i}
Demo	{"command":"Force_Position_Move","pose":[100000,200000,30000,400,500,600],"sensor":0,"mode":0, "dir":0, "force":15}
	Explain：Force-position hybrid control compensation by passthrough the target pose. Target pose：x：0.1m，y:0.2m，z：0.03m，Rx：0.4rad， Ry：0.5rad，Rz：0.6rad Z-direction compensation using one-axis forces along the base coordinate system with force control as Fz：1.5N
	{"command":"Force_Position_Move","joint":[1000,2000,3000,4000,5000,6000],"sensor":0,"mode":0, "dir":0, "force":15}
	Explain：Force-position hybrid control compensation by passthrough the target pose. The target angle of Joint 1~6：1°，2°，3°，4°，5°，6° Z-direction compensation using one-axis forces along the base coordinate system with force control as Fz：1.5N
Return	Planning success - returns the current angle of each joint and the force or torque of the force control method used {s:s,s:[i,i,i,i,i,i],s:i}
	{"state":"Force_Position_State","joint":[10,20,30,40,50,60], force:-15} The angle of Joint 1~6 is 0.01°~0.06° and the force or torque applied is -1.5
	Planning failed - return error message, {s:s,s:b}
	{"command":"Force_Position_Move", "set_state":false}
Comment	The starting point of the transmission/passthrough must be the current position of the robot arm, otherwise the force control compensation may fail or the robot arm may not be able to move.

(3) Stop the passthrough force-position hybrid control mode

Functionality	Turn off the underlying force-position hybrid control compensation mode. This command must be issued to turn off the function after completing the passthrough trajectory.

Argument description	Stop_Force_Position_Move
Command format	{s:s}
Demo	{"command":"Stop_Force_Position_Move"}
Explanation	N/A
Return	{s:s,s:b}
	{"command":"Stop_Force_Position_Move","set_state":true}
	True： Setup succeeded. False： Setup failed.

5. Lift (optional)

(1) Speed open-loop control

Functionality	To open-loop control the speed of the lift.
Argument description	set_lift_speed: Set lifting speed.
Command format	{s:s, s:i}
Demo	{"command":"set_lift_speed", "speed":50}
Explanation	speed-percentage, -100~100. Speed<0: Lift downward. Speed>0: Lift upward. Speed=0: Stop lifting.
Return	{" command ":"set_lift_speed ", "set_state":true}Setup succeeded. {" command ":"set_lift_speed ", "set_state":false}Setup failed.

(2) Position closed-loop control

Functionality	To closed-loop control the position of the lift.
Argument description	set_lift_height: Set lifting height.
Command format	{s:s, s:i}
Demo	{"command":"set_lift_ height ", "height":1000, "speed":50}
Explanation	height-target height, unit:mm, range: 0~2600. speed-speed percentage, 1~100.
Return	{"state":"current_trajectory_state","trajectory_state":true}Setup succeeded. {"state":"current_trajectory_state","trajectory_state":false}Setup failed.

(3) Get the status of the lift

Functionality	To get the status of the lift.
Argument description	get_lift_state: Get lift status.
Command format	{s:s}
Demo	{"command":" get_lift_state"}
Return	{"state":"lift_state", "height":1000, "current":500, "err_flag":0,"mode":1}

	{" command ":"set_lift_speed ", "set_state":false}Setup failed.
Explanation	height: height of the lift, unit: mm, accuracy:1mm, range:0~2300. current: current of the lift, unit: mA, accuracy: 1mA. err_flag: Lifting drive error code. The type of error code refers to joint error code. Mode is the current lift status, 0-idle, 1-positive direction velocity movement, 2-positive direction position movement, 3-negative direction velocity movement, 4-negative direction position movement.

6. Robot Arm Joints Calibration

6.1 Control of Calibration

(1) Start calibration

Functionality	To start joints calibration.
Argument description	
Command format	{s:s}
Demo	{"command":"start_calibrate"}
	Explain： the command to start joints calibration
Return	{"command":"start_calibrate","calibrate_state":true}Setup succeeded. {"command":"start_calibrate","calibrate_state":false}Setup failed.

(2) Stop calibration

Functionality	To stop joints calibration.
Argument description	
Command format	{s:s}
Demo	{"command":"stop_calibrate"}
	Explain： the command to stop joints calibration
Return	{"state":"stop_calibrate","calibrate_state":true}Setup succeeded. {"state":"stop_calibrate","calibrate_state":false}Setup failed.

6.2 Status of Calibration

(1) Query the calibration status

Functionality	To query the joints calibration status.
Argument description	
Command format	{s:s}
Demo	{"command":"get_calibrate_state"}

Return	{"state":"calibrate_state","joint_state":[5,0,0,0,0,0],"err":[0, 0,0,0,0,0,],”time”:0} 0-not calibrated, 1-in calibration, 2-calibrated, 3-joint stucked,4-calibration timeout, 5-error. 0x0001FOC frequency too high 0x0010 start-up failure 0x0100 temperature sensor error 0x0002 overvoltage 0x0020 code plate error 0x0200 position overrun error 0x0004 undervoltage 0x0040 overcurrent 0x0400 DRV8320 error 0x0008 overtemperature 0x0080 software error 0x0800 position tracking error overrun 0x1000 Current detection error time: Calibration duration (s)

7. Online Programming

7.1 File transfer

(1)Preparing for file delivery

Functionality	Starts preparing for file delivery.
Argument description	project_name:File name ;file_size:File size,plan_speed:Planning speed.
Command format	{s:s,s:s,s:i}
Demo	{"command":"run_project","project_name":"XXX","file_size":2048,"plan_speed":50}
Explanation	file_size: File size, unit:byte. plan-speed: Planning speed proportional coefficient.
Return	{"command":"run_project","project_state":true} Ready.

(2)In process of sending(start with the second one)

Functionality	The robot arm returns a frame of this instruction every time it receives 2k data.
Argument description	
Command format	{s:s,s:i}
Demo	{}
Explanation	Send residual data,Max 2k.
Return	{"command":"conduct_project","project_conduct":true}Successful reception

(3)Send the return value of checkout

Functionality	After receiving the last frame data, the controller checks and returns a successful or failed state.
Argument description	project_state:results of check;err_line:Number of wrong rows
Command format	{s:s,s:b}或者{s:s,s:b,s:i}
Demo	File transfer completed
Explanation	
Return	{"command":"download_project","project_state":true}Successful verification, instruction bullet box prompt {"command":"download_project","project_state":false,"err_line":60} Verification failure， err_line is Number of wrong rows,verification, instruction bullet box prompt，and mark the trajectory red。If err_line is 0, the length of the calibration data is not correct.

(4)Change speed coefficients during planning

Functionality	Change speed coefficients during planning.
Argument description	speed:Speed.
Command format	{s:s,s:i}
Demo	{"command":"plan_speed","speed":50}
Explanation	speed：The speed coefficient of the current progress bar.
Return	{"command":"plan_speed","plan_state":true}Set speed coefficient successfully.

7.2 Prompt box

(1)Pop-up window prompts

Functionality	File tree popup prompt. This instruction is sent by the controller to the instructor, and the return value is sent by the instructor to the controller.
Argument description	content：The location of the file tree where the pop-up window prompt instruction is located.
Command format	{s:s,s:i}
Demo	{"state":"popup","content":1}
Explanation	Execute the first popover in the file tree.

Return	{"command":"popup","popup_result":true}Continue.

7.3 Drag teaching

(1)Get the trajectory recorded by dragging

Functionality	Get the trajectory recorded by dragging,it can be used after the drag teaching is finished.
Argument description	save_trajectory：Save trajectory.
Command format	{s:s,s:s}
Demo	{"command":"save_trajectory", "trajectory_name":"XXX"}
Explanation	trajectory_name：The name of the track to save（arbitrarily）.
Return	{"point":" [30971, 56885, -3416, 76201, 47121, -4845]} Return all the point of drag teaching.

Appendix：Error Code

1. System Error Code

#	Error Code (Hexadecimal)	Content
1	0x0000	System normal
2	0x1001	Abnormal joint communication
3	0x1002	Target angle exceeds limit
4	0x1003	The place is not reachable and is a singularity
5	0x1004	Real-time kernel communication error
6	0x1005	Joint communication bus error
7	0x1006	Planning level kernel error
8	0x1007	Joint overspeed
9	0x1008	End interface board cannot be connected
10	0x1009	Reserved, undefined
11	0x100A	Reserved, undefined
12	0x100B	Joint brake not open
13	0x100C	Over speed during drag teaching
14	0x100D	robot collision
15	0x100E	No current operation coordinate system

16	0x100F	No current tool coordinate system
17	0x1010	The joint is disenabled

2. Joint Error Code

#	Error Code (Hexadecimal)	Content
1	0x0000	Joint normal
2	0x0001	FOC error
3	0x0002	Overvoltage
4	0x0004	Undervoltage
5	0x0008	Over temperature
6	0x0010	Start-up failure
7	0x0020	Initial positioning failure
8	0x0040	Overcurrent
9	0x0080	Software error
10	0x0100	Temperature sensor error
11	0x0200	Position sensor error
12	0x0400	Drive chip error
13	0x0800	Position tracking error
14	0x1000	Current detection error
15	0x2000	Brake opening failure
16	0x8000	Abnormal temperature
17	0xF000	Communication frame loss