1. Overview

CR robots now support two remote control modes: **remote I/O mode** and **remote Modbus mode**. For details about the control mode, please see Software Usage Instructions > Settings > Remote Control in *Dobot-CR-Series Robot-App-User-Guide-V3.7*.

The above two modes are mainly for the **remote control of running scripts**. As the communication based on TCP/IP has high reliability, strong practicability and high performance with low cost, many industrial automation projects have a wide demand for control robots that support TCP/IP protocol. So CR robots, designed on the basis of TCP/IP protocol, provide rich interfaces for interaction with external devices.

2. Message Format

According to the design, CR robots will open 29999 and 30003 server ports; 29999 server port (hereinafter referred to as Dashboard port) is responsible for receiving some simple instructions by sending and receiving one by one. That is, **after receiving the agreed message format from the client, the Dashboard port will give feedback to the client.** 30003 server port (hereinafter referred to as the real-time feedback port) **feeds back the robot information every 8 ms. It only receives the agreed message format from the client but does not give feedback.**

2.1 Format for sending messages

`Message name(Param1,Param2,Param3……Paramn)`

The message format is shown as above. It consists of a message name, parameters in a bracket. Each parameter is separated by an English comma “,”. A complete message ends up with right parenthesis.

Both message commands and message responses are in ASCII format (string).

2.2 Format for Feedback

2.2.1 Feedback when receiving successfully

`"Message name(Param1,Param2,Param3……Paramn)"`

It returns the message name, as shown above.

2.2.2 Feedback when failing to receive

`"could not understand:'Message name(Param1,Param2,Param3……Paramn)'"`

It returns "could not understand:'Message name(Param1,Param2,Param3……Paramn)'", as shown above.

3. Communication Protocol—Dashboard Port

The upper computer can directly send some simple instructions to the robot through 29999 port. These instructions are defined by CR, and these functions are called Dashboard. The table below is the Dashboard instruction list. The Dashboard commands can be used to control the robot, including enabling/disabling and resetting the robot.

Commands	Description
EnableRobot	enable the robot
DisableRobot	disable the robot
ClearError	clear the error of the robot
ResetRobot	the robot stops its current action, and receives enabling again
SpeedFactor	set the global speed ratio
User	select the identified user coordinate system
Tool	select the identified tool coordinate system
RobotMode	mode of robot
PayLoad	set the current load
DO	set the status of digital output port
DOExecute	set the status of digital output port (immediate command)
ToolDO	set the status of digital output port of the tool
ToolDOExecute	set the status of digital output port of the tool (immediate command)
AO	set the voltage of analog output port of controller
AOExecute	set the voltage of analog output port of controller (immediate command)
AccJ	set the joint acceleration rate. This command is valid only when the motion mode is MovJ, MovJIO, MovJR, JointMovJ
AccL	set the Cartesian acceleration rate. This command is valid only when the motion mode is MovL, MovLIO, MovLR, Jump, Arc, Circle
SpeedJ	set the joint velocity rate. This command is valid only when the motion mode is MovJ, MovJIO, MovJR, JointMovJ
SpeedL	set the Cartesian velocity rate. This command is valid only when the motion mode is MovL, MovLIO, MovLR, Jump, Arc, Circle
Arch	set the index of arc parameters (StartHeight, zLimit, EndHeight) in the Jump mode
CP	set the continuous path rate during movement
LimZ	set the maximum lifting height in Jump mode
SetArmOrientation	set the orientation of the arm
PowerOn	Power on the robot
RunScript	run the script
StopScript	stop the script
PauseScript	pause the script

Commands	Description
ContinueScript	continue the script
GetHoldRegs	read the holding register value
SetHoldRegs	write in the holding register

3.1 EnableRobot

- Function: EnableRobot()
- Description: enable the robot
- Parameters: null
- Supporting port: 29999
- Example

```
EnableRobot()
```

3.2 DisableRobot

- Function: DisableRobot()
- Description: disable the robot
- Parameters: None
- Supporting port: 29999
- Example

```
DisableRobot()
```

3.3 ClearError

- Function: ClearError()
- Description: clear the error of the robot
- Parameters: null
- Supporting port: 29999
- Example

```
ClearError()
```

3.4 ResetRobot

- Function: ResetRobot()
- Description: stop the robot
- Parameters: None
- Supporting port: 29999
- Example

```
ResetRobot()
```

3.5 SpeedFactor

- Function: SpeedFactor(ratio)
- Description: set the global speed ratio
- Parameters:

Parameter	Type	Description
ratio	int	speed ratio, range: 0~100, exclusive of 0 and 100

- Supporting port: 29999
- Example

```
SpeedFactor(80)
```

3.6 User

- Function: User(index)
- Description: select the identified user coordinate system
- Parameters:

Parameter	Type	Description
index	int	select the identified user coordinate system, range: 0~9

- Supporting port: 29999
- Example

```
User(1)
```

3.7 Tool

- Function: Tool(index)
- Description: select the identified tool coordinate system
- Parameters:

Parameter	Type	Description
index	int	select the identified tool coordinate system, range: 0~9

- Supporting port: 29999
- Example

```
Tool(1)
```

3.8 RobotMode

- Function: RobotMode()
- Description: mode of robot
- Parameters: None
- Supporting port: 29999
- Example

```
RobotMode()
```

- Return:

Mode	Description	Note
-1	ROBOT_MODE_NO_CONTROLLER	No controller
0	ROBOT_MODE_DISCONNECTED	Disconnect
1	ROBOT_MODE_CONFIRM_SAFETY	Configure safety parameters
2	ROBOT_MODE_BOOTING	Start
3	ROBOT_MODE_POWER_OFF	Power off
4	ROBOT_MODE_POWER_ON	Power on
5	ROBOT_MODE_IDLE	Idle
6	ROBOT_MODE_BACKDRIVE	Drag
7	ROBOT_MODE_RUNNING	Run
8	ROBOT_MODE_UPDATING_FIRMWAREu	Update firmware
9	ROBOT_MODE_ERROR	Alarm

3.9 PayLoad

- Function: PayLoad(weight,inertia)
- Description: set the current load
- Parameters:

Parameter	Type	Description
weight	double	load weight kg
inertia	double	load inertia kgm^2

- Supporting port: 29999
- Example

```
PayLoad(3,0.4)
```

3.10 DO

- Function: DO(index,0/1)

- Description: set the status of digital output port (queue command)
- Parameters:

Parameter	Type	Description
index	int	digital output index, range: 1~24
0/1	bool	status of the digital output port. 1: High level; 0: Low level

- Supporting port: 29999
- Example

```
DO(1,1)
```

3.11 DOExecute

- Function: DOExecute(index,0/1)
- Description: set the status of digital output port (immediate command)
- Parameters:

Parameter	Type	Description
index	int	digital output index, range: 1~16
0/1	boo	status of the digital output port. 1: High level; 0: Low level

- Supporting port: 29999
- Example

```
DOExecute(1,1)
```

3.12 ToolDO

- Function: ToolDO(index,0/1)
- Description: set the status of digital output port of the tool (queue command)
- Parameters:

Parameter	Type	Description
index	int	digital output index, range: 1 or 2
0/1	bool	status of digital output port. 1: high level, 0: low level

- Supporting port: 29999
- Example

```
ToolDO(1,1)
```

3.13 ToolDOExecute

- Function: ToolDOExecute(index,0/1)

- Description: set the status of digital output port of the tool (immediate command)
- Parameters:

Parameter	Type	Description
index	int	digital output index, range: 1 or 2
0/1	bool	status of the digital output port. 1: high level; 0: low level

- Supporting port: 29999
- Example

```
ToolDOExecute(1,1)
```

3.14 AO

- Function: AO(index,value)
- Description: set the voltage of analog output port of controller (queue command)
- Parameters:

Parameter	Type	Description
index	int	analog output index, range: 1 or 2
value	double	voltage of corresponding index, range: 0~10

- Supporting port: 29999
- Example

```
AO(1,2)
```

3.15 AOExecute

- Function: AOExecute(index,value)
- Description: set the voltage of analog output port of controller (immediate command)
- Parameters:

Parameter	Type	Description
index	int	analog output index, range: 1 or 2
value	double	voltage of corresponding index, range: 0~10

- Supporting port: 29999
- Example

```
AOExecute(1,2)
```

3.16 AccJ

- Function: AccJ(R)

- Description: set the joint acceleration rate. This command is valid only when the motion mode is MovJ, MovJIO, MovJR, JointMovJ
- Parameters:

Parameter	Type	Description
R	int	joint acceleration rate, range: 1~100

- Supporting port: 29999
- Example

```
AccJ(50)
```

3.17 AccL

- Function: AccL(R)
- Description: set the Cartesian acceleration rate. This command is valid only when the motion mode is MovL, MovLIO, MovLR, Jump, Arc, Circle
- Parameters:

Parameter	Type	Description
R	int	Cartesian acceleration rate, range: 1~100

- Supporting port: 29999
- Example

```
AccL(50)
```

3.18 SpeedJ

- Function: SpeedJ(R)
- Description: set the joint velocity rate. This command is valid only when the motion mode is MovJ, MovJIO, MovJR, JointMovJ
- Parameters:

Parameter	Type	Description
R	int	joint velocity rate, range: 1~100

- Supporting port: 29999
- Example

```
SpeedJ(50)
```

3.19 SpeedL

- Function: SpeedL(R)
- Description: set the Cartesian velocity rate. This command is valid only when the motion mode is MovL, MovLIO, MovLR, Jump, Arc, Circle

- Parameters:

Parameter	Type	Description
R	int	Cartesian velocity rate, range: 1~100

- Supporting port: 29999
- Example

```
SpeedL(50)
```

3.20 Arch

- Function: Arch(Index)
- Description: set the index of arc parameters (StartHeight, zLimit, EndHeight) in the Jump mode
- Parameters:

Parameter	Type	Description
Index	int	arc parameters index, range: 0~9

- Supporting port: 29999
- Example

```
Arch(1)
```

3.21 CP

- Function: CP(R)
- Description: set CP rate. CP means continuous path, that is, when the robot arm reaches the end point from the starting point through the intermediate point, it passes through the intermediate point in a right angle or in a curve. This command is invalid for Jump mode.
- Parameters:

Parameter	Type	Description
R	int	continuous path rate, range: 1~100

- Supporting port: 29999
- Example

```
CP(50)
```

3.22 LimZ

- Function: LimZ(zValue)
- Description: set the maximum lifting height in Jump mode
- Parameters:

Parameter	Type	Description
zValue	int	maximum lifting height which cannot exceed the Z-axis limiting position of the robot

- Supporting port: 29999
- Example

```
LimZ(80)
```

3.23 SetArmOrientation

- Function: SetArmOrientation(LorR,UorD,ForN,Config6)
- Description: set the orientation of the arm
- Parameters:

Parameter	Type	Description
LorR	int	Arm direction: forward/backward (1/-1)
UorD	int	Arm direction: up the elbow/down the elbow (1/-1)
ForN	int	Whether the wrist is reversed (1/-1)
Config6	int	Sixth axis Angle sign

- Supporting port: 29999
- Example

```
SetArmOrientation(1,1,-1,1)
```

3.24 PowerOn

- Function: PowerOn()
- Description: Power on the robot
- Parameters: None
- Supporting port: 29999

  **Note: Once the robot is powered on, you can enable the robot after about 10 seconds.

- Example

```
PowerOn()
```

3.25 RunScript

- Function: RunScript(projectName)
- Description: run the script
- Parameters:

Parameter	Type	Description
projectName	string	script name

- Supporting port: 29999
- Example

```
RunScript(demo)
```

3.26 StopScript

- Function: StopScript()
- Description: stop the script
- Parameters: None
- Supporting port: 29999
- Example

```
StopScript()
```

3.27 PauseScript

- Function: PauseScript()
- Description: pause the script
- Parameters: None
- Supporting port: 29999
- Example

```
PauseScript()
```

3.28 ContinueScript

- Function: ContinueScript()
- Description: continue the script
- Parameters: None
- Supporting port: 29999
- Example

```
ContinueScript()
```

3.29 GetHoldRegs

- Function: GetHoldRegs(id,*addr*, *count*,type)
- Description: read the holding register value
- Parameters:

Parameter	Type	Description
id	int	device ID of slave station, supporting at most five devices, range: 0~4. You should set it to 0 when accessing the internal slave of controller
addr	int	starting address of the holding registers. range: 3095~4095
count	int	read the specified amount of data of type, range: 1~16
type	string	data type: If it is empty, read 16-bit unsigned integer (two bytes, occupy one register) “U16”: read 16-bit unsigned integer (two bytes, occupy one register) “U32”: read 32-bit unsigned integer (four bytes, occupy two registers) “F32”: read 32-bit single-precision floating-point number (four bytes, occupy two registers) “F64”: read 64-bit double-precision floating-point number (eight bytes, occupy four registers)

- Supporting port: 29999
- Example

  ```
  data= GetHoldRegs(0,3095,1)
  ```

  Read a 16-bit unsigned integer starting at address 3095.

3.30 SetHoldRegs

- Function: SetHoldRegs(id,*addr*, *count*,*table*,type)
- Description: write in the holding register
- Parameters:

Parameter	Type	Description
id	int	device ID of slave station, supporting at most five devices, range: 0~4. You should set it to 0 when accessing the internal slave of controller
addr	int	starting address of the holding registers. range: 3095~4095
count	int	write the specified amount of data of type, range: 1~16
table	int	holding register value
type	string	data type: If it is empty, read 16-bit unsigned integer (two bytes, occupy one register) “U16”：read 16-bit unsigned integer (two bytes, occupy one register) “U32”：read 32-bit unsigned integer (four bytes, occupy two registers) “F32”：read 32-bit single-precision floating-point number (four bytes, occupy two registers) “F64”：read 64-bit double-precision floating-point number (eight bytes, occupy four registers)

- Supporting port: 29999
- Example

```
SetHoldRegs(0,3095,2,{6000,300}, “U16”)
```

Write two 16-bit unsigned integers: 6000 and 300, starting at address 3095.

4. Communication Protocol—Real- time Feedback Port

30003 port (real-time feedback port) is not only used to send the agreed motion-related protocols, but has other functions. The client can receive the robot information every 20ms, as shown in the following table. Each packet received through the real-time feedback port has 1440 bytes, which are arranged in a standard format. The following table shows the order of the bytes.

Meaning	Type	Number of values	Size in bytes	Byte position value	Notes
Message Size	unsigned short	1	2	0000 ~ 0001	Total message length in bytes
	unsigned short	3	6	0002 ~ 0007	Reserved bits
Digital input bits	double	1	8	0008 ~ 0015	Current state of the digital inputs
Digital outputs	double	1	8	0016 ~ 0023	digital output
Robot Mode	double	1	8	0024 ~ 0031	Robot mode
Controller Timer	double	1	8	0032 ~ 0039	Controller realtime thread execution time
Time	double	1	8	0040 ~ 0047	Time elapsed since the controller was started
test_value	double	1	8	0048 ~ 0055	Standard values for memory structure test: 0x0123 4567 89AB CDEF
Safety Mode	double	1	8	0056 ~ 0063	Safety mode
Speed scaling	double	1	8	0064 ~ 0071	Speed scaling of the trajectory limiter
Linear momentum norm	double	1	8	0072 ~ 0079	Norm of Cartesian linear momentum
V main	double	1	8	0080 ~ 0087	Masterboard: Main voltage
V robot	double	1	8	0088 ~ 0095	Masterboard: Robot voltage (48V)
I robot	double	1	8	0096 ~ 0103	Masterboard: Robot current
Program state	double	1	8	0104 ~ 0111	Program state
Safety Status	double	1	8	0112 ~ 0119	Safety status

Meaning	Type	Number of values	Size in bytes	Byte position value	Notes
Tool Accelerometer values	double	3	24	0120 ~ 0143	Tool x,y and z accelerometer values
Elbow position	double	3	24	0144 ~ 0167	Elbow position
Elbow velocity	double	3	24	0168 ~ 0191	Elbow velocity
q target	double	6	48	0192 ~ 0239	Target joint positions
qd target	double	6	48	0240 ~ 0287	Target joint velocities
qdd target	double	6	48	0288 ~ 0335	Target joint accelerations
I target	double	6	48	0336 ~ 0383	Target joint currents
M target	double	6	48	0384 ~ 0431	Target joint moments (torques)
q actual	double	6	48	0432 ~ 0479	Actual joint positions
qd actual	double	6	48	0480 ~ 0527	Actual joint velocities
I actual	double	6	48	0528 ~ 0575	Actual joint currents
I control	double	6	48	0576 ~ 0623	Joint control currents(temporally replaced by 0)
Tool vector actual	double	6	48	0624 ~ 0671	Actual Cartesian coordinates of the tool: (x,y,z,rx,ry,rz), where rx, ry and rz is a rotation vector representation of the tool orientation
TCP speed actual	double	6	48	0672 ~ 0719	Actual speed of the tool given in Cartesian coordinates
TCP force	double	6	48	0720 ~ 0767	Generalised forces in the TCP

Meaning	Type	Number of values	Size in bytes	Byte position value	Notes
Tool vector target	double	6	48	0768 ~ 0815	Target Cartesian coordinates of the tool: (x,y,z,rx,ry,rz), where rx, ry and rz is a rotation vector representation of the tool orientation
TCP speed target	double	6	48	0816 ~ 0863	Target speed of the tool given in Cartesian coordinates
Motor temperatures	double	6	48	0864 ~ 0911	Temperature of each joint in degrees celsius
Joint Modes	double	6	48	0912 ~ 0959	Joint control modes
V actual	double	6	48	960 ~ 1007	Actual joint voltages
	double	54	432	1008 ~ 1439	Reserved bits
TOTAL		183	1440		183 values in a 1440-byte package

Robot Mode returns the mode of robo as follows:

Mode	Description	Note
-1	ROBOT_MODE_NO_CONTROLLER	No controller
0	ROBOT_MODE_DISCONNECTED	Disconnect
1	ROBOT_MODE_CONFIRM_SAFETY	Configure safety parameter
2	ROBOT_MODE_BOOTING	Start
3	ROBOT_MODE_POWER_OFF	Power off
4	ROBOT_MODE_POWER_ON	Power on
5	ROBOT_MODE_IDLE	Idle
6	ROBOT_MODE_BACKDRIVE	Drag
7	ROBOT_MODE_RUNNING	Run
8	ROBOT_MODE_UPDATING_FIRMWARE	Update firmware
9	ROBOT_MODE_ERROR	Alarm

- Description：

- □ If the robot is powered off, the mode is 3.
- □ If the robot is powered on but not enabled, the mode is 4;
- □ If the robot is enabled successfully, the mode is 5.
- □ If the robot enters drag mode (enabled state), the mode is 6;
- □ If the robot moves, the mode is 7;
- □ Priority: Power on < Idle < Drag = Run < Power off < Alarm
- □ Alarm is the top priority. When other modes exist simultaneously, if there is an alarm, the mode is set to 9 first;

The following table shows the motion command protocols supported by the real-time feedback port. The real-time feedback port only receives commands but does not give feedback.

Command	Description
MovJ	point to point movement, the target point is Cartesian point
MovL	linear movement, the target point is Cartesian point
JointMovJ	point to point movement, the target point is joint point
Jump	Jump movement, only supports Cartesian points
RelMovJ	move to the Cartesian offset position in a point-to-point mode
RelMovL	move to the Cartesian offset position in a straight line
MovLIO	set the status of digital output port in straight line movement (can set several groups)
MovJIO	set the status of digital output port in point-to-point movement, and the target point is Cartesian point
Arc	arc movement, needs to combine with other motion commands
Circle	circular movement, needs to combine with other motion commands
ServoJ	dynamic following command based on joint space
ServoP	dynamic following command based on Cartesian space

4.1 MovJ

- Function: MovJ(X,Y,Z,A,B,C)
- Description: point to point movement, the target point is Cartesian point
- Parameters:

Parameter	Type	Description
X	double	X-axis coordinates, unit: mm
Y	double	Y-axis coordinates, unit: mm
Z	double	Z-axis coordinates, unit: mm
A	double	A-axis coordinates, unit: °
B	double	B-axis coordinates, unit: °
C	double	C-axis coordinates, unit: °

- Supporting port: 30003
- Example

```
MovJ(-500,100,200,150,0,90)
```

4.2 MovL

- Function: MovL(X,Y,Z,A,B,C)
- Description: linear movement, the target point is Cartesian point
- Parameters:

Parameter	Type	Description
X	double	X-axis coordinates, unit: mm
Y	double	Y-axis coordinates, unit: mm
Z	double	Z-axis coordinates, unit: mm
A	double	A-axis coordinates, unit: °
B	double	B-axis coordinates, unit: °
C	double	C-axis coordinates, unit: °

- Supporting port: 30003
- Example

```
MovL(-500,100,200,150,0,90)
```

4.3 JointMovJ

- Function: JointMovJ(J1,J2,J3,J4,J5,J6)
- Description: point to point movement, the target point is joint point
- Parameters:

Parameter	Type	Description
J1	double	J1 coordinates, unit: °
J2	double	J2 coordinates, unit: °
J3	double	J3 coordinates, unit: °
J4	double	J4 coordinates, unit: °
J5	double	J5 coordinates, unit: °
J6	double	J6 coordinates, unit: °

- Supporting port: 30003
- Example

```
JointMovJ(0,0,-90,0,90,0)
```

4.4 Jump

To be determined

4.5 RelMovJ

- Function: RelMovJ(offset1,offset2,offset3,offset4,offset5,offset6)
- Description: move to the Cartesian offset position in a point-to-point mode
- Parameters:

Parameter	Type	Description
offset1	double	J1-axis offset, unit: °
offset2	double	J2-axis offset, unit: °
offset3	double	J3-axis offset, unit: °
offset4	double	J4-axis offset, unit: °
offset5	double	J5-axis offset, unit: °
offset6	double	J6-axis offset, unit: °

- Supporting port: 30003
- Example

```
RelMovJ(10,10,10,10,10,10)
```

4.6 RelMovL

- Function: RelMovL(offsetX,offsetY,offsetZ)
- Description: move to the Cartesian offset position in a straight line
- Parameters:

Parameter	Type	Description
offsetX	double	X-axis offset in the Cartesian coordinate system; unit: mm
offsetY	double	Y-axis offset in the Cartesian coordinate system; unit: mm
offsetZ	double	Z-axis offset in the Cartesian coordinate system; unit: mm

- Supporting port: 30003
- Example

```
RelMovL(10,10,10)
```

4.7 MovLIO

- Function: MovLIO(X,Y,Z,A,B,C,{Mode,Distance,Index,Status},...,{Mode,Distance,Index,Status})
- Description: set the status of digital output port in straight line movement, and the target point is Cartesian point
- Parameters:

Parameter	Type	Description
X	double	X-axis coordinates, unit: mm
Y	double	Y-axis coordinates, unit: mm
Z	double	Z-axis coordinates, unit: mm
A	double	A-axis coordinates, unit: °
B	double	B-axis coordinates, unit: °
C	double	C-axis coordinates, unit: °
Mode	int	mode of Distance. 0: distance percentage; 1: distance away from the starting point or target point
Distance	int	move specified distance. If Mode is 0, Distance refers to the distance percentage between the starting point and target point; range: 0~100. If Distance value is positive, it refers to the distance away from the starting point; If Distance value is negative, it refers to the distance away from the target point
Index	int	digital output index, range: 1~24
Status	int	digital output status, range: 0 or 1

- Supporting port: 30003
- Example

```
MovLIO(-500,100,200,150,0,90,{0,50,1,0})
```

4.8 MovJIO

- Function: MovJIO(X,Y,Z,A,B,C,{Mode,Distance,Index,Status},...,{Mode,Distance,Index,Status})
- Description: set the status of digital output port in point-to-point movement, and the target point is Cartesian point
- Parameters:

Parameter	Type	Description
X	double	X-axis coordinates, unit: mm
Y	double	Y-axis coordinates, unit: mm
Z	double	Z-axis coordinates, unit: mm
A	double	A-axis coordinates, unit: °
B	double	B-axis coordinates, unit: °
C	double	C-axis coordinates, unit: °
Mode	int	mode of Distance. 0: distance percentage; 1: distance away from the starting point or target point
Distance	int	move specified distance. If Mode is 0, Distance refers to the distance percentage between the starting point and target point; range: 0~100. If Distance value is positive, it refers to the distance away from the starting point; If Distance value is negative, it refers to the distance away from the target point
Index	int	digital output index, range: 1~24
Status	int	digital output status, range: 0 or 1

- Supporting port: 30003
- Example

```
MovJIO(-500,100,200,150,0,90,{0,50,1,0})
```

4.9 Arc

- Function: Arc(X1,Y1,Z1,A1,B1,C1,X2,Y2,Z2,A2,B2,C2)
- Description: move from the current position to a target position in an arc interpolated mode under the Cartesian coordinate system
  This command needs to combine with other motion commands to obtain the starting point of an arc trajectory
- Parameters:

Parameter	Type	Description
X1	double	X1-axis coordinates of arc center point, unit: mm
Y1	double	Y1-axis coordinates of arc center point, unit: mm
Z1	double	Z1-axis coordinates of arc center point, unit: mm
A1	double	A1-axis coordinates of arc center point, unit: °
B1	double	B1-axis coordinates of arc center point, unit: °
C1	double	C1-axis coordinates of arc center point, unit: °
X2	double	X2-axis coordinates of arc ending point, unit: mm
Y2	double	Y2-axis coordinates of arc ending point, unit: mm
Z2	double	Z2-axis coordinates of arc ending point, unit: mm
A2	double	A2-axis coordinates of arc ending point, unit: °
B2	double	B2-axis coordinates of arc ending point, unit: °
C2	double	C2-axis coordinates of arc ending point, unit: °

- Supporting port: 30003

4.10 Circle

- Function: Circle(count,X1,Y1,Z1,A1,B1,C1,X2,Y2,Z2,A2,B2,C2)
- Description: circular movement. This command needs to combine with other motion commands
- Parameters:

Parameter	Type	Description
count	int	number of circles
X1	double	X1-axis coordinates, unit: mm
Y1	double	Y1-axis coordinates, unit: mm
Z1	double	Z1-axis coordinates, unit: mm
A1	double	A1-axis coordinates, unit: °
B1	double	B1-axis coordinates, unit: °
C1	double	C1-axis coordinates, unit: °
X2	double	X2-axis coordinates, unit: mm
Y2	double	Y2-axis coordinates, unit: mm
Z2	double	Z2-axis coordinates, unit: mm
A2	double	A2-axis coordinates, unit: °
B2	double	B2-axis coordinates, unit: °
C2	double	C2-axis coordinates, unit: °

- Supporting port: 30003

4.11 ServoJ

- Function: ServoJ(J11,J12,J13,J14,J15,J16)
- Description: dynamic following command based on joint space
- Parameters:

Parameter	Type	Description
J11	double	J11 coordinates of P1, unit: °
J12	double	J12 coordinates of P1, unit: °
J13	double	J13 coordinates of P1, unit: °
J14	double	J14 coordinates of P1, unit: °
J15	double	J15 coordinates of P1, unit: °
J16	double	J16 coordinates of P1, unit: °

- Supporting port: 30003
- Example

```
ServoJ(0,0,-90,0,90,0)
```

4.12 ServoP

- Function: ServoP(X1,Y1,Z1,A1,B1,C1)

- Description: dynamic following command based on Cartesian space
- Parameters:

Parameter	Type	Description
X1	double	X1-axis coordinates, unit: mm
Y1	double	Y1-axis coordinates, unit: mm
Z1	dou	Z1-axis coordinates, unit: mm
A1	double	A1-axis coordinates, unit: °
B1	double	B1-axis coordinates, unit: °
C1	double	C1-axis coordinates, unit: °

- Supporting port: 30003
- Example

```
ServoP(-500,100,200,150,0,90)
```