

Table of Contents

Introduction	1.1
1 Motion Commands	1.2
1.1 Point to point, the target point is Cartesian point	1.2.1
1.2 Point to point, the target point is joint point	1.2.2
1.3 Linear movement	1.2.3
1.4 Arc movement	1.2.4
1.5 Jump movement	1.2.5
1.6 Circle movement	1.2.6
1.7 Cartesian point offset	1.2.7
1.8 Joint point offset	1.2.8
1.9 Move to the Cartesian offset position in a straight line	1.2.9
1.10 Move to the Cartesian offset position in a point-to-point mode	1.2.10
1.11 Move to the joint offset position in a point-to-point mode	1.2.11
1.12 Block the program from executing queue instructions	1.2.12
1.13 Go movement in parallel with output	1.2.13
1.14 Move movement in parallel with output	1.2.14
1.15 MoveJ movement in parallel with output	1.2.15
1.16 Check the status of trajectory after Move command is executed	1.2.16
1.17 Check whether the robot moves to the target point	1.2.17
2 Motion Parameter Commands	1.3
2.1 Set the acceleration rate of Go, Jump, or MoveJ	1.3.1
2.2 Set the acceleration rate of Move, Jump, Arc3, Circle3 and MoveR	1.3.2
2.3 Set the velocity rate of Go, MoveJ, GoR and MoveJR	1.3.3
2.4 Set the velocity rate of Move, Jump, Arc3, Circle3 and MoveR	1.3.4
2.5 Set the index of arc parameters in the Jump mode	1.3.5
2.6 CP	1.3.6
2.7 Set the maximum lifting height in Jump mode	1.3.7
2.8 Set global speed ratio	1.3.8
2.9 Set the posture speed rate of Move, Jump, Arc3, Circle3 and MoveR	1.3.9
2.10 Set the posture acceleration rate of Move, Jump, Arc3, Circle3 and MoveR	1.3.10
3 Six-axis Force Sensor Commands	1.4
3.1 Homing six-axis force sensor	1.4.1
3.2 Spiral motion to find the hole position	1.4.2
3.3 Rotation motion to find the hole position	1.4.3
3.4 Linear jack movement	1.4.4

4 Input&Output Commands	1.5
4.1 DI	1.5.1
4.2 DO	1.5.2
4.3 DOExecute	1.5.3
4.4 ToolDI	1.5.4
4.5 ToolDO	1.5.5
4.6 ToolDOExecute	1.5.6
4.7 ToolAnalogMode	1.5.7
4.8 ToolAI	1.5.8
4.9 AI	1.5.9
4.10 AO	1.5.10
4.11 AOExecute	1.5.11
4.12 WaitDI	1.5.12
5 Program Managing Commands	1.6
5.1 Motion command waiting	1.6.1
5.2 Set delay time	1.6.2
5.3 Pause program operation	1.6.3
5.4 Start timing	1.6.4
5.5 Stop timing	1.6.5
5.6 Get current time	1.6.6
5.7 Print	1.6.7
6 Pose Getting Commands	1.7
6.1 Get Cartesian coordinates	1.7.1
6.2 Get joint coordinates	1.7.2
7 TCP Commands	1.8
7.1 Create TCP	1.8.1
7.2 Establish TCP connection	1.8.2
7.3 Receive TCP data	1.8.3
7.4 Send TCP data	1.8.4
7.5 Close TCP	1.8.5
7.6 Get data of a single point	1.8.6
7.7 Get data of multiple points	1.8.7
8 UDP Commands	1.9
8.1 Create UDP	1.9.1
8.2 Receive UDP data	1.9.2
8.3 Send UDP data	1.9.3
9 Modbus Commands	1.10
9.1 Create Modbus master station	1.10.1
9.2 Disconnect with Modbus slave	1.10.2

9.3	Read the value from the Modbus slave coil register address	1.10.3
9.4	Set the coil register in the Modbus slave	1.10.4
9.5	Read the value from the Modbus slave discrete register address	1.10.5
9.6	Read the value from the Modbus slave input register address	1.10.6
9.7	Read the value from the Modbus slave holding register address	1.10.7
9.8	Set the holding register in the Modbus slave	1.10.8
10	Tool Commands	1.11
10.1	Set power state of the tool	1.11.1
10.2	Set tool baud rate	1.11.2
11	Coordinate System Commands	1.12
11.1	Modify user coordinate system	1.12.1
11.2	Calculate user coordinate system	1.12.2
11.3	Modify tool coordinate system	1.12.3
11.4	Calculate tool coordinate system	1.12.4
12	Encoder Commands	1.13
12.1	Set the current value of the encoder	1.13.1
12.2	Get the current position of the encoder	1.13.2
13	Trajectory Playback Commands	1.14
13.1	Trajectory fitting	1.14.1
13.2	Get the first point in trajectory fitting	1.14.2
13.3	Trajectory playback	1.14.3
13.4	Get the first point in trajectory playback	1.14.4
14	Load Commands	1.15
14.1	Set the current load	1.15.1
14.2	Switch the parameter-setting status of the load	1.15.2
15	Pallet Commands	1.16
15.1	Instantiate matrix pallet	1.16.1
15.2	Set the next stack index to be operated	1.16.2
15.3	Get the current operated stack index	1.16.3
15.4	Set the next pallet layer index to be operated	1.16.4
15.5	Get the current pallet layer index	1.16.5
15.6	Reset pallet	1.16.6
15.7	Check whether the stack assembly or dismantling is completed	1.16.7
15.8	Release palletizing instance	1.16.8
15.9	The robot moves from the current position to the first stack position as the configured stack assembly path	1.16.9
15.10	The robot moves from the current position to the safe point as the configured stack dismantling path	1.16.10
16	Conveyer Tracking Commands	1.17

16.1 Set conveyor number to create a tracing queue	1.17.1
16.2 Obtain status of the object	1.17.2
16.3 Set X-axis, Y-axis offset under the set User coordinate system	1.17.3
16.4 Set time compensation	1.17.4
16.5 Synchronize the specified conveyor	1.17.5
16.6 Stop synchronous conveyor	1.17.6
17 Other Commands	1.18
17.1 Set the on-off state of safeskin	1.18.1
17.2 Set the state of obstacle avoidance of safeskin	1.18.2
17.3 Set collision level	1.18.3

Introduction

CC series controller encapsulates the robot dedicated API commands for programming with Lua language. This section describes commonly used commands for reference.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Motion Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Point to point, the target point is Cartesian point

- Function:

```
Go(P," User=1 Tool=2 CP=1 Speed=50 Accel=20 SYNC=1 ")
```

- Description: move from the current position to a target position in a point-to-point mode under the Cartesian coordinate system
- Required parameter: P: target point, which is user-defined or obtained from the TeachPoint page. Only Cartesian coordinate points are supported
- Optional parameter:
 - CP: whether to set continuous path function, range: 0~100
 - Speed: velocity rate, range: 1~100
 - Accel: acceleration rate, range: 1~100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Example

```
Go(P1)
```

The robot moves to P1 with the default setting.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Point to point, the target point is joint point

- Function:

```
MoveJ(P,"CP=1 Speed=50 Accel=20 SYNC=1")
```

- Description: move from the current position to a target position in a point-to-point motion under the Joint coordinate system
- Required parameter: P: joint angle of the target point, which cannot be obtained from the TeachPoint page. You need to define the joint coordinate point before calling this command
- Optional parameter:
 - CP: whether to set continuous path function. Value range: 0~100
 - Speed: velocity rate. Value range: 1~100
 - Accel: acceleration rate. Value range: 1~100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Example

```
local P = {joint={0,-0.0674194,0,0,0,0}}  
MoveJ(P)
```

Define the joint coordinate point P. Move the robot to P.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:58

Linear movement

- Function:

```
Move(P," User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

- Description: Move from the current position to a target position in a straight line under the Cartesian coordinate system
- Required parameter: P: target point, which is user-defined or obtained from the TeachPoint page. Only Cartesian coordinate points are supported
- Optional parameter:
 - CP: whether to set continuous path function, range: 0~100
 - SpeedS: velocity rate, range: 1~100
 - AccelS: acceleration rate, range: 1~100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Example

```
Move(P1)
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Arc movement

- Function:

```
Arc3(P1,P2, " User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

- 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
- Required parameter:
 - P1: middle point, which is user-defined or obtained from the TeachPoint page. Only Cartesian coordinate points are supported
 - P2: end point, which is user-defined or obtained from the TeachPoint page. Only Cartesian coordinate points are supported
- Optional parameter:
 - CP: whether to set continuous path function. range: 0~ 100
 - SpeedS: velocity rate. range: 1~100
 - AccelS: acceleration rate. range: 1~100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Example

```
While true do  
  Go(P1)  
  Arc3(P2,P3)  
end
```

The robot cycles from P1 to P3 via P2 in the arc interpolated mode.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Jump movement

- Function:

```
Jump(P," User=1 Tool=2 SpeedS=50 AccelS=20 Start=10 ZLimit=80 End=50 SYNC=1")
```

- Description: The robot moves from the current position to a target position in the Move mode. The trajectory looks like a door. This command should be used combined with other commands
- Required parameter: P: target point, which is user-defined or obtained from the TeachPoint page. Only Cartesian coordinate points are supported. Also, the target point cannot be higher than ZLimit to avoid an alarm about JUMP parameter error
- Optional parameter:
 - SpeedS: Velocity rate. Value range: 1~100
 - AccelS: Acceleration rate. Value range: 1~100
 - Arch: Arch index. Value range: 0~9
 - Start: Lifting height
 - ZLimit: Maximum lifting height
 - End: Dropping height
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Example

```
Go(P6)  
Jump(P5,"Start=10 ZLimit=600 End=10")
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-17 12:41:48

Circle movement

- Function:

```
Circle3(P1,P2, Count, "User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

- Description: move from the current position to a target position in a circular 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

- Required parameter:
 - P1: middle point, which is user-defined or obtained from the TeachPoint page. Only Cartesian coordinate points are supported
 - P2: end point, which is user-defined or obtained from the TeachPoint page. Only Cartesian coordinate points are supported
 - Count: number of circles, range: 1 - 999
- Optional parameter:
 - CP: Whether to set continuous path function. Value range: 0 - 100
 - SpeedS: Velocity rate. Value range: 1 - 100
 - AccelS: Acceleration rate. Value range: 1 - 100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Example

```
Go(P1)  
Circle3(P2,P3,1)
```

The robot cycles from P1 to P2, and then to P3 in the circular interpolated mode.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Cartesian point offset

- Function:

```
RP(P1, {OffsetX, OffsetY, OffsetZ})
```

- Description: set the X-axis, Y-axis, Z-axis offset under the Cartesian coordinate system to return a new Cartesian coordinate point The robot can move to this point in all motion commands except MoveJ
- Optional parameter:
 - P1: current Cartesian coordinate point, which is user-defined or obtained from the TeachPoint page. Only Cartesian coordinate points are supported
 - OffsetX, OffsetY, OffsetZ: X-axis, Y-axis, Z-axis offset in the Cartesian coordinate system; unit: mm
- Return: Cartesian coordinate point
- Example

```
P2=RP(P1, {50,10,32})  
Move(P2) or Move(RP(P1, {50,10,32}))
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Joint point offset

- Function:

```
RJ(P1, {Offset1, Offset2, Offset3, Offset4, Offset5, Offset6})
```

- Description: set the joint offset in the Joint coordinate system to return a new joint coordinate point

The robot can move to this point only in MoveJ command

- Parameter:

- P1: current joint point, which cannot be obtained from the TeachPoint page. You need to define the joint coordinate point
- Offset1~Offset6: J1~J6 axes offset; unit: °

- Return: joint coordinate point

- Example

```
local P1 = {joint={0, -0.0674194, 0, 0, 0, 0}}  
P2=RJ(P1, {60, 50, 32, 30, 25, 30})  
MoveJ(P2)或 MoveJ(RJ(P1, {60, 50, 32, 30, 25, 30}))
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Move to the Cartesian offset position in a straight line

- Function:

```
MoveR({OffsetX, OffsetY, OffsetZ}, "User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

- Description: move from the current position to the offset position in a straight line under the Cartesian coordinate system
- Required parameter: OffsetX, OffsetY, OffsetZ: X-axis, Y-axis, Z-axis offset in the Cartesian coordinate system; unit: mm
- Optional parameter:
 - CP: whether to set continuous path function, range: 0~100
 - SpeedS: velocity rate, range: 1~100
 - AccelS: acceleration rate, range: 1~100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Example

```
Go(P1)  
MoveR({20,20,20}, "AccelS=100 SpeedS=100 CP=100")
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Move to the Cartesian offset position in a point-to-point mode

- Function:

```
GoR({OffsetX, OffsetY, OffsetZ}, " User=1 Tool=2 CP=1 Speed=50 Accel=20 SYNC=1 ")
```

- Description: move from the current position to the offset position in a point-to-point mode under the Cartesian coordinate system
- Required parameter: OffsetX, OffsetY, OffsetZ: X-axis, Y-axis, Z-axis offset in the Cartesian coordinate system; unit: mm
- Optional parameter:
 - CP: whether to set continuous path function, range: 0~100
 - Speed: velocity rate, range: 1~100
 - Accel: acceleration rate, range: 1~100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely

- Example

```
Go(P1)  
GoR({10,10,10}, "Accel=100 Speed=100 CP=100")
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Move to the joint offset position in a point-to-point mode

- Function:

```
MoveJR({Offset1, Offset2, Offset3, Offset4, Offset5, Offset6}, "CP=1 Speed=50 Accel=20 SYNC=1")
```

- Description: move from the current position to the offset position in a point-to-point motion under the Joint coordinate system
- Required parameter: Offset1 - Offset6: J1 - J6 axes offset; unit: °
- Optional parameter:
 - CP: whether to set continuous path function, range: 0~100
 - Speed: velocity rate, range: 1~100
 - Accel: acceleration rate, range: 1~100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Example

```
Go(P1)  
MoveJR({20, 20, 10, 0, 10, 0}, "SYNC=1")
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Block the program from executing queue instructions

- Function:

```
Sync()
```

- Description: block the program from executing queue instructions, and return after all the instructions are executed
- Parameter: null

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Go movement in parallel with output

- Function:

```
GoIO(P, { {Mode, Distance, Index, Status},{Mode, Distance, Index, Status}...}, "ARM=Left User=1 Tool=2 CP=1 Speed=50 Accel=20 SYNC=1")
```

- Description: set the status of digital output port when the robot is moving under Go mode
- Required parameter:
 - P: target point, which is user-defined or obtained from the TeachPoint page. Only Cartesian coordinate points are supported.
 - Mode: set the mode of Distance. 0: distance percentage; 1: distance away from the starting point or target point
 - Distance: move specified distance
 - If Mode is 0, Distance refers to the distance percentage between the starting point and the target point. range: 0~100
 - If Mode is 1, Distance refers to the distance away from the starting point or target point
 - If the Distance value is positive, it refers to the distance away from the starting point
 - If the Distance value is negative, it refers to the distance away from the target point
 - Index: digital output index, range: 1~24
 - Status: digital output status, range: 0 or 1
- Optional parameter:
 - ARM: direction of robot arm. If the robot is a horizontal multi-joint robot (SCARA four-axis), set it to Left or Right; if it is a vertical multi-joint robot (six-axis), the parameter is invalid
 - User: user coordinate system, range: 0~9
 - Tool: tool coordinate system, range: 0~9
 - CP: whether to set continuous path function, range: 0~100
 - Speed: velocity rate, range: 1~100
 - Accel: acceleration rate, range: 1~100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Return: null
- Example

```
GoIO(P1, {0, 10, 2, 1})
```

The robot arm moves to P1 with the default setting. When it moves 10% distance, set digital output 2 to ON.

Move movement in parallel with output

- Function:

```
MoveIO(P, { {Mode, Distance, Index, Status},{Mode, Distance, Index, Status}...}, "ARM=Left User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

- Description: set the status of digital output port when the robot is moving under Move mode
- Required parameter:
 - P: target point, which is user-defined or obtained from the TeachPoint page. Only Cartesian coordinate points are supported.
 - Mode: set the mode of Distance. 0: distance percentage; 1: distance away from the starting point or target point
 - Distance: move specified distance
 - If Mode is 0, Distance refers to the distance percentage between the starting point and the target point. range: 0~100
 - If Mode is 1, Distance refers to the distance away from the starting point or target point
 - If the Distance value is positive, it refers to the distance away from the starting point
 - If the Distance value is negative, it refers to the distance away from the target point
 - Index: digital output index, range: 1~24
 - Status: digital output status, range: 0 or 1
- Optional parameter:
 - ARM: direction of robot arm. If the robot is a horizontal multi-joint robot () set it to Left or Right; if it is a vertical multi-joint robot (six-axis), the parameter is invalid
 - CP: whether to set continuous path function, range: 0~100
 - Speed: velocity rate, range: 1~100
 - Accel: acceleration rate, range: 1~100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Return: null
- Example

```
MoveIO(P1, {0, 10, 2, 1})
```

The robot moves to P1 with the default setting. When it moves 10% distance, set the digital output 2 to ON.

MoveJ movement in parallel with output

- Function:

```
MoveJIO(P, { {Mode, Distance, Index, Status},{Mode, Distance, Index, Status}...}, "CP=1 Speed=50 Accel=20 SYNC=1")
```

- Description: set the status of digital output port when the robot is moving under MoveJ mode
- Required parameter:
 - P: joint angle of the target point, which cannot be obtained from the TeachPoint page. You need to define the joint coordinate point before calling this command
 - Mode: mode of Distance. 0: distance percentage; 1: distance away from the starting point or target point
 - Distance: move specified distance

If Mode is 0, Distance refers to the distance percentage between the starting point and target point; range: 0~100

If Mode is 1, Distance refers to the distance away from the starting point or target point

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: digital output index, range: 1~24
 - Status: digital output status, range: 1~24
- Optional parameter:
 - CP: whether to set continuous path function, range: 0~100
 - Speed: velocity rate, range: 1~100
 - Accel: acceleration rate, range: 1~100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Return: null
- Example

```
MoveJIO (P1, {0, 10, 2, 1})
```

The robot moves to P1 with the default setting. When it moves 10% distance away from P1, set the digital output 2 to ON.

Check the status of trajectory after Move command is executed

- Function:

```
CheckMove(P," User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
```

- Description: check the status of trajectory after Move command is executed
- Required parameter: P, target point, which can be obtained from Teachpoint page, or self-defined. Only points under the Cartesian coordinate system are supported.
- Optional parameter:
 - CP: Whether to set continuous path function. Value range: 0~100
 - SpeedS: Velocity rate. Value range: 1~100
 - AccelS: Acceleration rate. Value range: 1~100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Return: trajectory status of motion commands
 - 0: no error
 - 16: The planned point is closed to the shoulder singularity point
 - 17: Inverse kinematics error with no solution
 - 18: Inverse kinematics error with result out of working area
 - 22: Arm orientation error
 - 26: The planned point is closed to the wrist singularity point
 - 27: The planned point is closed to the elbow singularity point
 - 29: Speed parameter is wrong
 - 32: Inverse kinematics error with shoulder singularity when robot moving
 - 33: Inverse kinematics error with no solution when robot moving
 - 34: Inverse kinematics error with result out of working area when robot moving
 - 35: Inverse kinematics with wrist singularity when robot moving
 - 36: Inverse kinematics with elbow singularity when robot moving
 - 37: The Joint angle is changed over 180 degree
- Example

```
local status=CheckMove(P1)
```

Detect the status of trajectory when the robot moves to P1 at the default speed.

Check whether the robot moves to the target point

- Function:

```
CheckGo(P," User=1 Tool=2 CP=1 Speed=50 Accel=20 SYNC=1")
```

- Description: check whether the robot moves to the target point
- Required parameter: P, target point, which is user-defined or obtained from the TeachPoint page. Only Cartesian coordinate points are supported.
- Optional parameter:
 - CP: Whether to set continuous path function, range: 0~100
 - Speed: Velocity rate, range: 1~100
 - Accel: Acceleration rate, range: 1~100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Return: trajectory status of motion commands
 - 0: no error
 - 16: The planned point is closed to the shoulder singularity point
 - 17: Inverse kinematics error with no solution
 - 18: Inverse kinematics error with result out of working area
 - 22: Arm orientation error
 - 26: The planned point is closed to the wrist singularity point
 - 27: The planned point is closed to the elbow singularity point
 - 29: Speed parameter is wrong
 - 32: Inverse kinematics error with shoulder singularity when robot moving
 - 33: Inverse kinematics error with no solution when robot moving
 - 34: Inverse kinematics error with result out of working area when robot moving
 - 35: Inverse kinematics with wrist singularity when robot moving
 - 36: Inverse kinematics with elbow singularity when robot moving
 - 37: The Joint angle is changed over 180 degree
- Example

```
local status=CheckGo (P1)
```

Detect the status of trajectory when the robot moves to P1 with the default setting.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Motion Parameter Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Set the acceleration rate of Go, Jump, or MoveJ

- Function:

```
Accel(R)
```

- Description: set the acceleration rate. This command is valid only when the motion mode is Go, Jump, or MoveJ
- Parameter:
 - R: percentage, range: 1~100
- Example

```
Accel(50)  
Go(P1)
```

The robot moves to P1 with 50% acceleration rate.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Set the acceleration rate of Move, Jump, Arc3, Circle3 and MoveR

- Function:

```
Acce1R(ratio)
```

- Description: set the acceleration rate of Move, Jump, Arc3, Circle3, MoveR.
- Parameter:
 - ratio: acceleration rate, range: 0~100, exclusive of 0 and 100
- Return: null
- Example

```
Acce1R (20)  
Move(P1)
```

The robot moves to P1 with the acceleration ratio of 20%.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Set the velocity rate of Go, MoveJ, GoR and MoveJR

- Function:

```
Speed(R)
```

- Description: set the velocity rate. This command is valid only when the motion mode is Go, MoveJ, GoR and MoveJR
- Parameter:
 - R: percentage, range: 1~100
- Example

```
Speed(20)  
Go(P1)
```

The robot moves to P1 with 20% velocity rate.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Set the velocity rate of Move, Jump, Arc3, Circle3 and MoveR

- Function:

```
SpeedS(R)
```

- Description: set the velocity ratio of Move, Jump, Arc3, Circle3 and MoveR
- Parameter:
 - R: percentage, range: 1~100
- Example

```
SpeedS(20)  
Move(P1)
```

The robot moves to P1 with 20% velocity ratio.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Set the index of arc parameters in the Jump mode

- Function:

```
Arch(Index)
```

- Description: set the index of arc parameters (StartHeight, zLimit, EndHeight) in the Jump mode
- Parameter: Index: arc parameters index, range: 0~9 This parameter needs to be set in Setting > PlaybackArch of the APP
- Example

```
Arch(1)  
Jump(P1)
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

CP

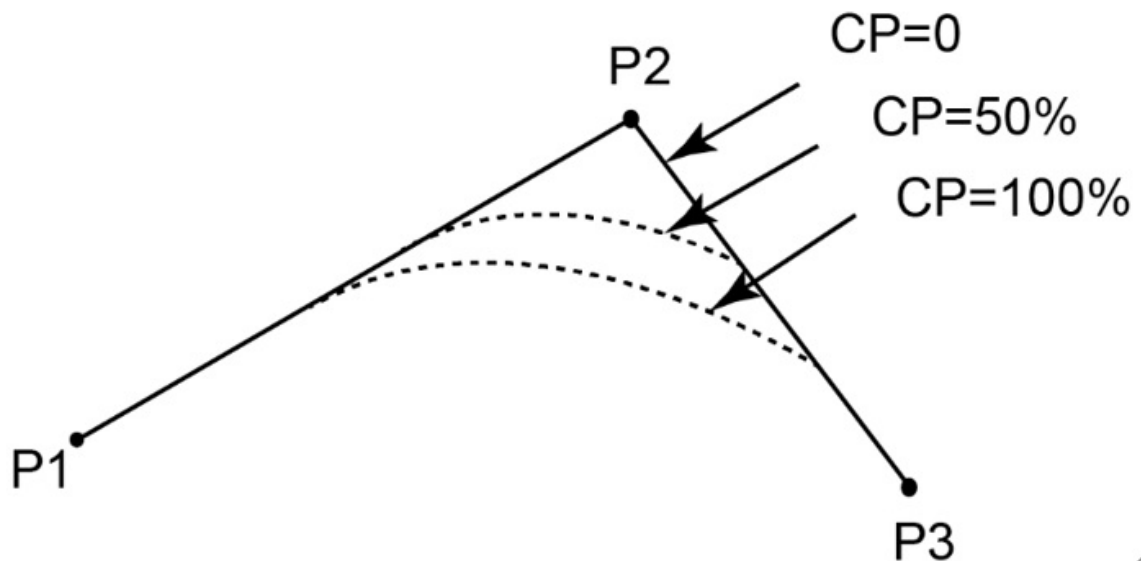
- Function:

```
CP(R)
```

- Description: set the continuous path rate. This command is valid only when the motion mode is Go, Move, Arc3, Circle3, or MoveJ
- Parameter: R: continuous path rate, range: 0~100 0 indicates that the continuous path function is locked
- Example

```
CP(50)  
Move(P1)  
Move(P2)  
Move(P3)
```

The robot moves from P1 to P2 with 50% continuous path rate.



Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Set the maximum lifting height in Jump mode

- Function:

```
LimZ(zValue)
```

- Description: set the maximum lifting height in Jump mode
- Parameter: zValue: maximum lifting height which cannot exceed the Z-axis limiting position of the robot
- Example

```
LimZ(80)  
Jump(P," Start=10 Zlimit=LimZ End=50")
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Set global speed ratio

- Function:

```
SpeedFactor(ratio)
```

- Description: set the global speed ratio
- Parameter:
 - ratio: speed ratio, range: 0~100, exclusive of 0 and 100
- Return: null
- Example

```
SpeedFactor (20)
```

Set the global speed ratio as 20%.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Set the posture speed rate of Move, Jump, Arc3, Circle3 and MoveR

- Function:

```
SpeedR(ratio)
```

- Description: set the posture speed rate of Move, Jump, Arc3, Circle3, MoveR
- Parameter:

```
* Ratio: speed ratio, range: 0~100, exclusive of 0 and 100
```

- Return: null
- Example

```
SpeedR (20)  
Move(P1)
```

The robot moves to P1 at the speed ratio of 20%.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Set the posture acceleration rate of Move, Jump, Arc3, Circle3 and MoveR

- Function:

```
Acce1R(ratio)
```

- Description: set the acceleration rate of Move, Jump, Arc3, Circle3, MoveR.
- Parameter:
 - ratio: acceleration rate, range: 0~100, exclusive of 0 and 100
- Return: null
- Example

```
Acce1R (20)  
Move(P1)
```

The robot moves to P1 with the acceleration ratio of 20%.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Six-axis Force Sensor Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Homing six-axis force sensor

- Function:

```
SixForceHome()
```

- Description: homing six-axis force sensor
- Parameter: null
- Example

```
SixForceHome()
```

Execute the command to home six-axis force sensor.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Spiral motion to find the hole position

- Function:

```
Spiral(P, User, Tool, Direction, SpeedC, Force, Insertion, Perturn, PeckMode, MaxValue)
```

- Description: the robot arm performs a spiral motion between the current position and the specified position to find the hole position. The specified point needs to be close to the hole position, which is the starting point for hole position exploration.
- Parameter:
 - P: specified position
 - User: user coordinate system, range: 0~9
 - Tool: tool coordinate system, range: 0~9
 - Direction: Jack direction (0: Forward, 1: Reverse)
 - SpeedC: Jack speed (mm/s)
 - Force: rotation threshold (N)
 - Insertion: jack threshold (N)
 - Perturn: spiral radius (mm)
 - PeckMode: point contact mode (ON/OFF)
 - MaxValue: maximum spiral radius (mm)
- Example

```
Spiral(P1,"User=1 Tool=2 Dirction=0 SpeedC=5 Force =10 Insertion=3 Perturn=0.7 PeckMode=OFF MaxValue =5")
```

Do a spiral motion between the current position and P1 to find the hole position. When the resistance in the direction of the jack is greater than the Force threshold, the robot performs a spiral motion to explore the hole position. When the resistance in the direction of the jack is less than the Insertion threshold, the robot moves in the direction of the jack to perform the jack work.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Rotation motion to find the hole position

- Function:

```
Rotation (P, User, Tool, Direction, SpeedC, Force, RotationSpeed, MaxTorque, PeckMode, MaxValue)
```

- Description: the robotic arm rotates between the current position and the specified position to find the hole position. The specified point needs to be close to the hole position, which is the starting point for hole position exploration.
- Parameter:
 - P: specified position
 - User: user coordinate system, range: 0~9
 - Tool: tool coordinate system, range: 0~9
 - Direction: Jack direction (0: Forward, 1: Reverse)
 - SpeedC: Jack speed (mm/s)
 - Force: rotation threshold (N)
 - RotationSpeed: rotation speed (°/s)
 - MaxTorque: maximum torque (Nm)
 - PeckMode: point contact mode (ON/OFF)
 - MaxValue: maximum spiral radius (mm)
- Example

```
Rotation (P1, "User=1 Tool=2 Dirction=0 SpeedC =5 Force =10 RotationSpeed=5 MaxTorque=1 PeckMode=OFF MaxValue =4 5")
```

Do a rotation between the current position and P1 to find the hole position. When the resistance in the direction of the jack is greater than the Force threshold, the robot performs a rotation to explore the hole position. When the resistance in the direction of the jack is less than the Force threshold, the robot moves in the direction of the jack to perform the jack work.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Linear jack movement

- Function:

```
Linear (User, Tool, Direction, SpeedC, Force, MaxValue)
```

- Description: the robot arm makes a linear jack movement in the direction of the hole
- Parameter:
 - User: user coordinate system, range: 0~9
 - Tool: tool coordinate system, range: 0~9
 - Direction: Jack direction (0: forward, 1: reverse)
 - SpeedC: Jack speed (mm/s)
 - Force: rotation threshold (N)
 - MaxValue: maximum spiral radius (mm)

- Example

```
Linear("User=1 Tool=2 Dirction=0 SpeedC =5 Force=10 MaxValue=45")
```

Do a linear jack movement at the current hole position. When the resistance in the insertion direction is greater than the Force threshold, the insertion is considered complete.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Input&Output Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

DI

- Function:

```
DI(index)
```

- Description: get the status of the digital input port
- Parameter:
 - index: digital input index, range: 1~32
- Return:
 - When an index is set in the DI function, DI(index) returns the status (ON/OFF) of this specified input port
 - When there is no index in the DI function, DI() returns the status of all the input ports, which are saved in a table For example, local di=(), the saving format is {num = 24 value = {0x55, 0xAA, 0x52}}, you can obtain the status of the specified input port through di.num and di.value[n]
- Example

```
if (DI(1))==ON then  
Move(P1)  
end
```

The robot moves to P1 when the status of the digital input port 1 is ON.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

DO

- Function:

```
DO(index,ON | OFF)
```

- Description: set the status of digital output port (queue command)
- Parameter:
 - index: digital output index, range: 1~24
 - ON/OFF: status of the digital output port. ON: High level; OFF: Low level
- Example

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

Set the status of the digital output port 1 to OFF.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

DOExecute

- Function:

```
DOExecute(index,ON | OFF)
```

- Description: Set the status of digital output port (Immediate command)
- Parameter:
 - index: digital output index, range: 1~16
 - ON/OFF: status of the digital output port. ON: High level; OFF: Low level
- Example

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

Set the status of the digital output port 1 to OFF.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

ToolDI

- Function:

```
ToolDI(index)
```

- Description: get the status of tool input port
- Parameter:
 - index: digital input index, range: 1 or 2
- Return:
 - status (ON/OFF) of the specified input port corresponding to the index type: number; value: 0: low level, 1: high level

If local di=(), the saving format is {num = 24 value = {0x55, 0xAA, 0x52}}, you can obtain the status of the specified input port through di.num and di.value[n]
- Example

```
if (ToolDI (1))==1 then  
Move(P1)  
end
```

The robot moves to P1 in a straight line when the status of the digital input port 1 of the tool is ON.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

TooIDO

- Function:

```
TooIDO(index, ON | OFF)
```

- Description: get the status of digital output port (arithmetic command)
- Parameter:
 - index: digital output index, range: 1 or 2
 - ON/OFF: status of digital output port. ON: high level, OFF: low level
- Return: null
- Example

```
TooIDO (1,OFF)
```

Set the status of the digital output port 1 to OFF.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

ToolDOExecute

- Function:

```
ToolDOExecute(index, ON | OFF)
```

- Description: set the status of digital output port (immediate command)
- Parameter:
 - index: digital output index, range: 1 or 2
 - ON/OFF: status of the digital output port. ON: high level; OFF: low level
- Return: null
- Example

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

Set the status of the digital output port 1 to OFF

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

ToolAnalogMode

- Function:

```
ToolAnalogMode(mode)
```

- Description: set the status of analog input port (Immediate command)

- Parameter:

- mode: mode of analog input port

00: default, 485 mode

10: current acquisition mode

11: 0~3.3V voltage input mode

12: 0~10V voltage input mode

- Return: null

- Example

```
ToolAnalogMode(11)
```

Set the status of analog input port as voltage input mode.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

ToolAI

- Function:

```
ToolAI(index)
```

- Description: get the voltage of analog input port of tool
- Parameter:
 - index: analog input index, range: 1 or 2

Note: Please set the mode before using, see ToolAnalogMode for details; the analog input port and RS485 communication port are multiplexed.

- Return: voltage of corresponding index
- Example

```
Voltage = ToolAI (1)
```

Get the voltage of input port 1 on the controller.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

AI

- Function:

```
AI(index)
```

- Description: get the voltage of analog input port of controller (immediate command)
- Parameter:
 - index: input index of controller, range: 1 or 2
- Return: voltage of corresponding index
- Example

```
Voltage = AI(1)
```

Get the voltage of analog input port 1 on the controller.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

AO

- Function:

```
AO(index,value)
```

- Description: set the voltage of analog output port of controller
- Parameter:
 - index: analog output index of controller, range: 1 or 2
 - value: voltage of corresponding index, range: 0~10, type: number
- Return: null
- Example

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

Set the voltage of analog output port of controller as 2V.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

AOExecute

- Function:

```
AOExecute(index,value)
```

- Description: set the voltage of analog output port of controller (immediate command)
- Parameter:
 - index: analog output index of controller, range: 1 or 2
 - value: voltage corresponding to index, range: 0~10, type: number
- Return:
 - index: analog output index of controller, range: 1 or 2
 - value: voltage corresponding to index, range: 0~10, type: number
- Example

```
TooIDOExecute (1,OFF)
```

Set the voltage of analog output port 1 of controller as 2V.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

WaitDI

- Function:

```
WaitDI(index, ON | OFF, period)
```

- Description: get the status of the digital input port. If the status is consistent with specified status, the program will continue to run. Otherwise, get the status of the digital input port at specified interval
- Parameter:
 - index: digital input index, range: 1~32
 - ON | OFF: status of digital input port, ON: high level; OFF: low level
 - period: interval, 50ms by default
- Return: null
- Example

```
WaitDI(1, ON)  
Move(P1)
```

Get the status of the digital input port 1 at the interval of 50ms. If the digital input port 1 is ON, the robot moves to P1 in a straight line.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-12 19:04:27

Program Managing Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Motion command waiting

- Function:

```
Wait(time)
```

- Description: set the delay time for robot motion commands
- Parameter:
 - time: delay time, unit: ms
- Example

```
Go(P1)  
Wait(1000)
```

Wait for 1000ms after the robot moves to P1.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Set delay time

- Function:

```
Sleep(time)
```

- Description: set the delay time for all commands

- Parameter:

- time: delay time, unit: ms

- Example




```
while true do
  Speed(100)
  Go(P1)
  sleep(3)
  Speed(100)
  Accel(40)
  Go(P2)
  sleep(3)
end
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Pause program operation

- Function:

```
Pause()
```

- Description: pause the running program When the program runs to this command, robot pauses running and the button  **Pause** on the APP turns into  **Resume** . If you want to run the robot, please click  **Resume**

- Parameter: null
- Example

```
while true
do
Go(P1)
Go(P2)
Pause()
Go(P3)
Go(P4)
end
```

The robot moves to P2 and then pauses running.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Start timing

- Function:

```
ResetElapsedTime()
```

- Description: start timing after all commands before this command are executed completely. The command should be used combined with ElapsedTime() command For example: calculate the time it takes to execute a piece of code
- Return: null
- Example

```
Go(P2, " Speed=100 Accel=100")
ResetElapsedTime()
for i=1,10 do
Jump(P1, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185")
Jump(P2, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185")
end
print (ElapsedTime())
Sleep(1000)
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Stop timing

- Function:

```
ElapsedTime()
```

- Description: stop timing and return the time difference. The command should be used combined with ResetElapsedTime() command
- Parameter: null
- Return: time difference., unit: ms
- Example

```
Go(P2, " Speed=100 Accel=100")
ResetElapsedTime()
for i=1,10 do
Jump(P1, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185")
Jump(P2, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185")
end
print (ElapsedTime())
Sleep(1000)
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Get current time

- Function:

```
Systemtime()
```

- Description: get the current time
- Parameter: null
- Return: current time
- Example

```
Go(P2, " Speed=100 Accel=100")
local time1=Systemtime()
for i=1,10 do
  Jump(P1, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185")
  Jump(P2, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185")
end
local time2=Systemtime()
local time = time2 - time1
Sleep(1000)
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Print

- Function:

```
print(value)
```

- Description: print the debug information
- Parameter:
 - value: data to be printed. supported data type: table, number, string and bool
- Return: null
- Example

```
print("hello world")
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Pose Getting Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Get Cartesian coordinates

- Function:

```
GetPose()
```

- Description: get the current pose of the robot under the Cartesian coordinate system If you have set the User or Tool coordinate system, the current pose is under the current User or Tool coordinate system
- Parameter: null
- Return: Cartesian coordinate of the current pose
- Example

```
local currentPose = GetPose()
--Get the current pose
local liftPose = {coordinate = {currentPose.coordinate[1], currentPose.coordinate[2], currentPose.coordinate[3]
},currentPose.coordinate[4] }, tool = currentPose.tool, user = currentPose.user}
-- Lift a certain height
Go(liftPose,"Speed=100 Accel=100")
Go(P1)
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Get joint coordinates

- Function:

```
GetAngle()
```

- Description: get the current pose of the robot under the Joint coordinate system
- Parameter: null
- Return: joint coordinate of the current pose
- Example

```
local armPose
local joint = GetAngle()
--Get the current pose
local liftPose = { joint = {joint.joint[1], joint.joint[2], joint.joint[3], joint.joint[4]}, tool = 0, user = 0}
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:58

TCP Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Create TCP

- Function:

```
err, socket = TCPCreate(isServer, IP, port)
```

- Description: Create a TCP network

Only a single connection is supported

- Parameter:

- isServer: whether to create a server. 0: Create a client; 1: Create a server
- IP: IP address of the server, which is in the same network segment of the client without conflict
- port: server port When the robot is set as a server, port cannot be set to 502 and 8080. Otherwise, it will be in conflict with the Modbus default port or the port used in the conveyor tracking application, causing the creation to fail

- Return:

- err: 0: TCP network is created successfully 1: TCP network failed to be created
- socket: socket object

- Example 1: TCP server demo

```
local ip="192.168.5.1" // IP address of the robot as a server
local port=6001 // Server port
local err=0
local socket=0
err, socket = TCPCreate(true, ip, port)
```

- Example 2: TCP client demo

```
``` local ip="192.168.5.25" // External equipment such as a camera is set as the server local port=6001 //
Server port local err=0 local socket=0 err, socket = TCPCreate(false, ip, port)
```

```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Establish TCP connection

- Function:

```
TCPStart(socket, timeout)
```

- Description: connect a client to a server with the TCP protocol
- Parameter:
 - socket: socket object
 - timeout: wait timeout. unit: s. If timeout is 0, the connection is still waiting. If not, the connection is exited after exceeding the timeout,
- Return:
 - 0: TCP connection is successful
 - 1: input parameters are incorrect
 - 2: socket object is not found
 - 3: timeout setting is incorrect
 - 4: If the robot is set as a client, it indicates that the connection is wrong. If the robot is set as a server, it indicates that receiving data is wrong
- Example 1: TCP server demo

```
local ip="192.168.5.1"                // IP address of the robot as a server
local port=6001                       // Server port
local err=0
local socket=0
err, socket = TCPCreate(true, ip, port)
if err == 0 then
err = TCPStart(socket, 0)
```

- Example 2: TCP client demo

```
local ip="192.168.5.25"               // External equipment such as a camera is set as the server
local port=6001                       // Server port
local err=0
local socket=0
err, socket = TCPCreate(false, ip, port)
if err == 0 then
err = TCPStart(socket, 0)
```

Receive TCP data

- Function:

```
err, Recbuf = TCPRead(socket, timeout, type)
```

- Description:

- Robot as a client receives data from a server
- Robot as a server receives data from a client

- Parameter:

- socket: socket object
- timeout: receiving timeout. unit: s. v
- type: buffer type. If type is not set, the buffer format of RecBuf is a table. If type is set to string, the buffer format is a string

- Return:

- err: 0: receiving data is successful 1: receiving data is failed
- Recbuf: data buffer

- Example 1: TCP server demo

```
local ip="192.168.5.1"                // IP address of the robot as a server
local port=6001                       // Server port
local err=0
local socket=0
err, socket = TCPCreate(true, ip, port)
if err == 0 then
err = TCPStart(socket, 0)
if err == 0 then
local RecBuf
while true do
TCPWrite(socket, "tcp server test")    // Server sends data to client
err, RecBuf = TCPRead(socket,0,"string") // Server receives the data from client
```

- Example 2: TCP client demo

```
local ip="192.168.5.25"              // External equipment such as a camera is set as the server
local port=6001                       // Server port
local err=0
local socket=0
err, socket = TCPCreate(false, ip, port)
if err == 0 then
err = TCPStart(socket, 0)
if err == 0 then
local RecBuf
while true do
TCPWrite(socket, "tcp client test")    // Client sends data to server
TCPWrite(socket, {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07})
err, RecBuf = TCPRead(socket, 0)      // Client receives data from server
```


Send TCP data

- Function:

```
TCPWrite(socket, buf, timeout)
```

- Description:

- The robot as a client sends data to a server
- The robot as a server sends data to a client

- Parameter:

- socket: socket object
- buf: data sent by the robot
- timeout: timeout. unit: s. If timeout is 0 or not set, this command is a block reading, namely, the program will not continue to run until sending data is completed. If timeout is not 0, after exceeding the timeout, the program will continue to run regardless of whether sending data is completed

- Return:

- 0: sending data is successful
- 1: sending data is failed

- Example 1: TCP server demo

```
local ip="192.168.5.1"                // IP address of the robot as a server
local port=6001                       // Server port
local err=0
local socket=0
err, socket = TCPCreate(true, ip, port)
if err == 0 then
err = TCPStart(socket, 0)
if err == 0 then
local RecBuf
while true do
TCPWrite(socket, "tcp server test")    // Server sends data to client
```

- Example 2: TCP client demo

```
local ip="192.168.5.25"               // External equipment such as a camera is set as the server
local port=6001                       // Server port
local err=0
local socket=0
err, socket = TCPCreate(false, ip, port)
if err == 0 then
err = TCPStart(socket, 0)
if err == 0 then
local RecBuf
while true do
TCPWrite(socket, "tcp client test")    // Client sends data to server
TCPWrite(socket, {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07})
err, RecBuf = TCPRead(socket, 0)      // Client receives data from server
```


Close TCP

- Function:

```
TCPDestroy(socket)
```

- Description: release a TCP network
- Parameter:
 - socket: socket object
- Return:
 - 0: releasing TCP is successful
 - 1: releasing TCP is failed
- Example 1: TCP server demo

```
local ip="192.168.5.1"                // IP address of the robot as a server
local port=6001                      // Server port
local err=0
local socket=0
err, socket = TCPCreate(true, ip, port)
if err == 0 then
err = TCPStart(socket, 0)
if err == 0 then
local RecBuf
while true do
TCPWrite(socket, "tcp server test")    // Server sends data to client
err, RecBuf = TCPRead(socket,0,"string") // Server receives the data from client
if err == 0 then
Go(P1)                                //Start to run motion commands after the server receives data
Go(P2)
print(buf)
else
print("Read error ".. err)
break
end
end
else
print("Create failed ".. err)
end
TCPDestroy(socket)
else
print("Create failed ".. err)
end
end
```

- Example 2: TCP client demo

```
local ip="192.168.5.25"              // External equipment such as a camera is set as the server
local port=6001                      // Server port
local err=0
local socket=0
err, socket = TCPCreate(false, ip, port)
if err == 0 then
err = TCPStart(socket, 0)
if err == 0 then
local RecBuf
```

```
while true do
  TCPWrite(socket, "tcp client test")           // Client sends data to server
  TCPWrite(socket, {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07})
  err, RecBuf = TCPRead(socket, 0)             // Client receives data from server
  if err == 0 then
    Go(P1)                                     // Start to run motion commands after the client receives the data
    Go(P2)
    print(buf)
  else
    print("Read error ".. err)
    break
  end
end
else
  print("Create failed ".. err)
end
TCPDestroy(socket)
else
  print("Create failed ".. err)
end
end
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Get data of a single point

- Function:

```
err, Recbuf = TCPReadVision(socket, timeout, type)
```

- Description: get the data of a single point
- Parameter:
 - socket: socket object
 - timeout: receiving timeout, unit: s. If timeout is 0 or is not set, this command is a block reading. If timeout is not 0, the program will continue to run after exceeding the timeout.
 - type: buffer type. If type is not set, the buffer format of RecBuf is a table. If type is set to string, the buffer format is a string
- Return:
 - err: 0: receiving data is successful 1: receiving data is failed
 - Recbuf: data buffer
- Example

```
local err, RecBuf = TCPReadVision (socket, timeout, "string");
```

The content of RecBuf is { coordinate={x, y, z, Rx, Ry, Rz}}.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-17 12:41:48

Get data of multiple points

- Function:

```
err, Recbuf = TCPReadMultiVision(socket, timeout, type)
```

- Description: get the data of multiple points
- Parameter:
 - socket: socket object
 - timeout: receiving timeout, unit: s. If timeout is 0 or is not set, this command is a block reading. If timeout is not 0, the program will continue to run after exceeding the timeout.
 - type: buffer type. If type is not set, the buffer format of RecBuf is a table. If type is set to string, the buffer format is a string
- Return:
 - err: 0:receiving data is successful 1:receiving data is failed
 - Recbuf: data buffer
- Example

```
local err, RecBuf = TCPReadMultiVision (socket, timeout, "string")
```

The content of RecBuf is { coordinate0={x, y, z, Rx, Ry, Rz}, coordinate1={x, y, z, Rx, Ry, Rz}}.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-17 13:09:55

UDP Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Create UDP

- Function:

```
err, socket = UDPCreate(isServer, IP, port)
```

- Description: create a UDP network Only a single connection is supported
- Parameter:
 - isServer: whether to create a server. 0: Create a client; 1: Create a server
 - IP: IP address of the server, which is in the same network segment of the client without conflict
 - port: server port When the robot is set as a server, port cannot be set to 502 or 8080. Otherwise, it will be in conflict with the Modbus default port or the port used in the conveyor tracking application, causing the creation to fail
- Return:
 - err: 0: The UDP network is created successfully 1: The UDP network failed to be created
 - socket: socket object
- Example 1: UDP server demo

```
local ip="192.168.5.1" // IP address of the robot as a server
local port=6201 // Server port
local err=0
local socket=0
err, socket = UDPCreate(true, ip, port)
```

- Example 2: UDP client demo

```
local ip="192.168.1.25" // IP address of the external equipment as a
server
local port=6200 // server port
local err=0
local socket=0
err, socket = UDPCreate(false, ip, port)
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Receive UDP data

- Function:

```
err, Recbuf = UDPRead(socket, timeout, type)
```

- Description:

- The robot as a client receives data from a server
- The robot as a server receives data from a client

- Parameter:

- socket: socket object
- timeout: Receiving timeout. Unit: s. If timeout is 0 or not set, this command is a block reading. Namely, the program will not continue to run until receiving data is complete. If not, after exceeding the timeout, the program will continue to run regardless of whether receiving data is complete
- type: Buffer type. If type is not set, the buffer format of RecBuf is a table. If type is set to string, the buffer format is a string

- Return:

- err: 0: Receiving data is successful 1: Receiving data is failed
- Recbuf: Data buffer

- Example 1: UDP server demo

```
local ip="192.168.5.1"                // IP address of the robot as a server
local port=6201                       // Server port
local err=0
local socket=0
err, socket = UDPCreate(true, ip, port)
if err == 0 then
  local RecBuf
  while true do
    UDPWrite(socket, "udp server test") // Server sends data to client
    err, RecBuf = UDPRead(socket, 0)    //Server receives the data from client
```

- Example 2: UDP client demo

```
local ip="192.168.1.25"              // IP address of the external equipment as a
server
local port=6200                       // server port
local err=0
local socket=0
err, socket = UDPCreate(false, ip, port)
if err == 0 then
  local RecBuf
  while true do
    UDPWrite(socket, "udp client test") // Client sends data to server
    UDPWrite(socket, {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07})
    err, RecBuf = UDPRead(socket, 0)    // Client receives the data from server
```


Send UDP data

- Function:

```
UDPWrite(socket, buf, timeout)
```

- Description:

- The robot as a client sends data to a server
- The robot as a server sends data to a client

- Parameter:

- socket: Socket object
- buf: Data sent by the robot
- timeout: Timeout. Unit: s. If timeout is 0 or not set, this command is a block reading. Namely, the program will not continue to run until sending data is complete. If not, after exceeding the timeout, the program will continue to run regardless of whether sending data is complete

- Return:

- 0: Sending data is successful
- 1: Sending data is failed

- Example 1: UDP server demo

```
local ip="192.168.5.1" // IP address of the robot as a server
local port=6201 // Server port
local err=0
local socket=0
err, socket = UDPCreate(true, ip, port)
if err == 0 then
local RecBuf
while true do
UDPWrite(socket, "udp server test") // Server sends data to client
```

- Example 2: UDP client demo

```
local ip="192.168.1.25" // IP address of the external equipment as a
server
local port=6200 // server port
local err=0
local socket=0
err, socket = UDPCreate(false, ip, port)
if err == 0 then
local RecBuf
while true do
UDPWrite(socket, "udp client test") // Client sends data to server
UDPWrite(socket, {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07})
```


Modbus Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Create Modbus master station

- Function:

```
ModbusCreate()
```

- Description: create Modbus master station, and establish connection with the slave station

- Parameter:

- IP: IP address of slave station
- port: slave station port
- slave_id: ID of slave station

- Return:

- err:
 - 0: Modbus master station is created successfully
 - 1: Modbus master station fails to be created
- id: device ID of slave station, supporting at most five devices, range: 0~4

Note: When ip, port, slave_id is void, or ip is 127.0.0.1 or 0.0.0.1, connect the Modbus slave station. For example, if you input any one of the following commands, it indicates connecting Modbus slave station.

- ModbusCreate()
- ModbusCreate("127.0.0.1")
- ModbusCreate("0.0.0.1")
- ModbusCreate("127.0.0.1",xxx,xxx) //xxx arbitrary value
- ModbusCreate("0.0.0.1",xxx,xxx) //xxx arbitrary value

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-17 12:41:48

Disconnect with Modbus slave

- Function:

```
ModbusClose()
```

- Description: disconnect with Modbus slave station
- Parameter:
 - id: device ID of slave station, supporting at most five devices, range: 0~4
- Return:
 - 0: Modbus master station is closed successfully
 - 1: Modbus master station fails to be closed
- Example

```
err, id = ModbusCreate(ip, port, slave_id)
if err == 0 then
  coils = {0, 1, 1, 1, 0}
  SetCoils(id, 1024, #coils, coils)
  ModbusClose(id)
else
  print("Create failed:", err)
end
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-12 19:04:27

Read the value from the Modbus slave coil register address

- Function:

```
GetCoils(id, addr, count)
```

- Description: read the coil value from the Modbus slave
- Parameter:
 - id: device ID of slave station, supporting at most five devices, range: 0~4
 - addr: starting address of the coils to read, range: 0~4095
 - count: number of the coils to read, range: 0 to 4096-addr
- Return: coil value stored in a table, where the first value in the table corresponds to the coil value at the starting address; data type: bit
- Example

```
Read 5 coils starting at address 0
Coils = GetCoils(id,0,5)
Return:
Coils={1,0,0,0,0}
As shown in Table 16.3, it indicates that the robot is in the starting state
```

| Coil register address (e.g.: PLC) | Coil register address (Robot system) | Data type | Description |
|-----------------------------------|--------------------------------------|-----------|----------------|
| 00001 | 0 | Bit | Start |
| 00002 | 1 | Bit | Pause |
| 00003 | 2 | Bit | Continue |
| 00004 | 3 | Bit | Stop |
| 00005 | 4 | Bit | Emergency stop |
| 00006 | 5 | Bit | Clear alarm |
| 00007~0999 | 6~998 | Bit | Reserved |
| 01001~04096 | 999~4095 | Bit | User-defined |

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Set the coil register in the Modbus slave

- Function:

```
SetCoils(id, addr, count, table)
```

- Description: set the address value of coil register in the Modbus slave This command is not supported when the coil register address is from 0 to 5
- Parameter:
 - id: device ID of slave station, supporting at most five devices, range: 0~4
 - Addr: starting address of the coils to set, range: 6 - 4095
 - count: number of the coils to set, range: 0 to 4096-addr
 - table: coil value, stored in a table, data type: bit
- Return: null
- Example

```
local Coils = {0,1,1,1,0}  
SetCoils(id, 1024, #coils, Coils)
```

Set 5 coils starting at address 1024.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Read the value from the Modbus slave discrete register address

- Function:

```
GetInBits(id, addr, count)
```

- Description: read the discrete input value from Modbus slave
- Parameter:
 - id: device ID of slave station, supporting at most five devices, range: 0~4
 - addr: starting address of the discrete inputs to read, range: 0~4095
 - count: number of the discrete inputs to read, range: 0 to 4096-addr
- Return: coil value stored in a table, where the first value in the table corresponds to the input register value at the starting address; data type: bit
- Example

```
Read 5 discrete inputs starting at address 0
inBits = GetInBits(id,0,5)
Return:
inBits = {0,0,0,1,0}
As shown in Table 17.1, it indicates the robot is in running state
```

| Coil register address (e.g.: PLC) | Coil register address (Robot system) | Data type | Description |
|-----------------------------------|--------------------------------------|-----------|----------------|
| 00001 | 0 | Bit | Start |
| 00002 | 1 | Bit | Pause |
| 00003 | 2 | Bit | Continue |
| 00004 | 3 | Bit | Stop |
| 00005 | 4 | Bit | Emergency stop |
| 00006 | 5 | Bit | Clear alarm |
| 00007~0999 | 6~998 | Bit | Reserved |
| 01001~04096 | 999~4095 | Bit | User-defined |

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Read the value from the Modbus slave input register address

- Function:

```
GetInRegs(id, addr, count, type)
```

- Description: read the input register value with the specified data type from the Modbus slave
- Parameter:
 - id: device ID of slave station, supporting at most five devices, range: 0~4
 - addr: starting address of the input registers, range: 0 - 4095
 - count: number of the input registers to read, range: 0 ~ 4096-addr
 - type: data type
 - 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)
- Return: value of input register stored in a table, where the first value in the table corresponds to the input register value at the starting address
- Example 1

```
data = GetInRegs(id,2048,1)
```

Read a 16-bit unsigned integer starting at address 2048.

- Example 2

```
data = GetInRegs(id, 2048, 1, "U32")
```

Read a 32-bit unsigned integer starting at address 2048.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Read the value from the Modbus slave holding register address

- Function:

```
GetHoldRegs(id, addr, count, type)
```

- Description: Read the holding register value from the Modbus slave according to the specified data type
- Parameter:
 - id: device ID of slave station, supporting at most five devices, range: 0~4
 - addr: starting address of the holding registers. Value range: 0 - 4095
 - count: number of the holding registers to read. Value range: 0 to 4096-addr
 - type: data type
 - 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)
- Return: coil value stored in a table, where the first value in the table corresponds to the input register value at the starting address
- Example 1:

```
data = GetHoldRegs(id,2048,1)
```

Read a 16-bit unsigned integer starting at address 2048.

- Example 2:

```
data = GetHoldRegs(id, 2048, 1, "U32")
```

Read a 32-bit unsigned integer starting at address 2048.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Set the holding register in the Modbus slave

- Function:

```
SetHoldRegs(id, addr, count, table, type)
```

- Description: set the holding register in the Modbus slave
- Parameter:
 - id: device ID of slave station, supporting at most five devices, range: 0~4
 - addr: starting address of the holding registers to set, range: 0 - 4095
 - count: number of the holding registers to set, range: 0 to 4096-addr
 - table: holding register value, stored in a table
 - type: datatype
 - 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)
- Return: null
- Example 1

```
local data = {6000}  
SetHoldRegs(id, 2048, #data, data, "U16")
```

Set a 16-bit unsigned integer starting at address 2048.

- Example 2

```
local data = {95.32105}  
SetHoldRegs(id, 2048, #data, data, "F64")
```

Set a 64-bit double-precision floating-point number starting at address 2048.

Tool Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

10.1 Set power state of the tool

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:34

Set tool baud rate

- Function:

```
SetToolBaudRate(baud)
```

- Description: set the baud rate of RS485 port of the tool
- Parameter:
 - baud: baud rate of RS485 port
- Return: null
- Example

```
SetToolBaudRate(115200)
```

Set the baud rate of RS485 port as 115200Hz.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Coordinate System Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:58

Modify user coordinate system

- Function:

```
SetUser(index, table)
```

- Description: modify user coordinate system
- Parameter:
 - index: index of coordinate system, range: 0~9 (0 is default coordinate system)
 - table: matrix for coordinate system, usually the return value of CalcUser()
- Return: null
- Example

```
Go(P1)
local u = {}
u = CalcUser(2,0,{10,10,0,0,0,0}) --User coordinate system 2 left multiplies {x,y,z,rx,ry,rz}; if the second parameter is 1, then user coordinate system 2 right multiplies {x,y,z,rx,ry,rz}.
    SetUser(2, u)    --Modify user coordinate system 2.
Go(P1,"User=2") --Use the modified user coordinate system 2.
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Calculate user coordinate system

- Function:

```
CalcUser(index,matrix_direction,table)
```

- Description: calculate user coordinate system
- Parameter:
 - Index: index of coordinate system, range: 0~9 (0 is default coordinate system)
 - matrix_direction: direction for calculation
 - 0: user coordinate system corresponding to the index left multiplies {x,y,z,rx,ry,rz}
 - 1: user coordinate system corresponding to the index right multiplies {x,y,z,rx,ry,rz}
 - Table: matrix for coordinate system {x,y,z,rx,ry,rz}
- Return: calculated matrix for coordinate system
- Example

```
Go(P1)
local u = {}
u = CalcUser(2,0,{10,10,0,0,0,0}) --User coordinate system 2 left multiplies {x,y,z,rx,ry,rz}; if the second parameter is 1, then user coordinate system 2 right multiplies {x,y,z,rx,ry,rz}.
  SetUser(2, u)    --Modify user coordinate system 2.
Go(P1,"User=2") --Use the modified user coordinate system 2.
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Modify tool coordinate system

- Function:

```
SetTool(index1,table)
```

- Description: modify tool coordinate system
- Parameter:
 - index1: index of coordinate system, range: 0~9 (0 is default coordinate system)
 - Table: matrix for coordinate system, usually the return value of CalcUser()
- Return: null
- Example

```
Go(P1)
local u = {}
u = CalcTool(2,0,{10,10,0,0,0,0}) --Tool coordinate system 2 left multiplies {x,y,z,rx,ry,rz}; if the second parameter is 1, then tool coordinate system 2 right multiplies {x,y,z,rx,ry,rz}.
    SetTool(2, u)    --Modify tool coordinate system 2.
Go(P1,"Tool=2") --Use the modified tool coordinate system 2.
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Calculate tool coordinate system

- Function:

```
CalcTool(index,matrix_direction,table)
```

- Description: calculate tool coordinate system
- Parameter:
 - Index: index of coordinate system, range: 0~9 (0 is default coordinate system)
 - matrix_direction: direction for calculation
 - 0: tool coordinate system corresponding to the index left multiplies {x,y,z,rx,ry,rz}
 - 1: tool coordinate system corresponding to the index right multiplies {x,y,z,rx,ry,rz}
 - Table: matrix for coordinate system {x,y,z,rx,ry,rz}
- Return: calculated matrix for coordinate system
- Example

```
Go(P1)
local u = {}
u = CalcTool(2,0,{10,10,0,0,0,0}) --Tool coordinate system 2 left multiplies {x,y,z,rx,ry,rz}; if the second parameter is 1, then tool coordinate system 2 right multiplies {x,y,z,rx,ry,rz}.
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 19:44:24

Encoder Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Set the current value of the encoder

- Function:

```
SetABZPPC(value)
```

- Description: set the current value of the encoder
- Parameter:
 - value: position of the encoder

The counter of the encoder is 32-bit

- Return: null
- Example

```
SetABZPPC(3000)
```

Set the current value of the encoder as 3000.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Get the current position of the encoder

- Function:

```
GetABZ()
```

- Description: get the current position of the encoder
- Parameter: null
- Return: current position of the encoder
- Example

```
local value=GetABZ()  
printf(value)
```

Get and print the current position of the encoder.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Trajectory Playback Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Trajectory fitting

- Function:

```
StartTrace(string, Option)
```

- Description: trajectory fitting
- Parameter:
 - String: name of the trace file (with the suffix)
 - option: settings for trajectory playback
 - CP: whether to set continuous path function, range: 0~100
 - SpeedS: velocity rate, range: 1~100
 - AccelS: acceleration rate, range: 1~100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Return: null
- Example

```
local string="demo.json"  
StartTrace(string, "CP=10 SpeedS=50 AccelS=20 SYNC=1")
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-13 16:57:15

Get the first point in trajectory fitting

- Function:

```
GetTraceStartPose(string)
```

- Description: get the first point of the trajectory
- Parameter:
 - string: name of the trajectory file (with the suffix)
- Return: coordinate value
- Example

```
local P1 = GetTraceStartPose(string)
```

Get the first point of the trajectory, and assign the value to P1.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-13 16:57:37

Trajectory playback

- Function:

```
StartPath(string, option)
```

- Description: trajectory playback
- Parameter:
 - String: name of the path file (with the suffix)
 - option: settings for trajectory playback
 - Multi: speed multiples, scope: 0.01~2.0
 - isConst: When isConst=1, it plays back at a constant speed, and the pauses and dead zones in the trajectory are removed
 - isCart: value is 0 or 1 (default value is 0)
 - 0: move joint points
 - 1: move Cartesian points
 - isRev: When isRev=1, the trajectory is reversed
 - isRel: When isRel=1, the overall trajectory offsets with the current point as the starting point
- Return: null
- Example

```
local string="demo.json"  
StartPath(string, "Multi=1 isConst=1 isCart=1 isRev=1 isRel=1")
```

The robot plays back at a one-multiple constant speed; at the same time, the trajectory is reversed,, and the overall trajectory offsets.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-13 16:58:57

Get the first point in trajectory playback

- Function:

```
GetPathStartPose(string)
```

- Description: get the first point of the trajectory
- Parameter:
 - string: name of the trajectory file (with the suffix)
- Example

```
local P1 = GetPathStartPose(string)
```

Get the first point of the trajectory, and assign the value to P1.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-13 16:57:56

Load Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Set the current load

- Function:

```
LoadSet(weight, inertia)
```

- Description: set the current load
- Parameter:
 - weight: load weight (kg)
 - inertia: load inertia (kgm²)
- Return: null
- Example

```
LoadSet(3, 0.4)
```

Set the load weight of the robot as 3 kg, and inertia as 0.4 kgm².

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Switch the parameter-setting status of the load

- Function:

```
LoadSwitch(status)
```

- Description: switch the parameter-setting status of the load
- Parameter:
 - status:
 - 0: lock the parameter setting of load
 - 1: unlock the parameter setting of load, which will enhance the sensitivity of collision detection
- Return: null
- Example

```
LoadSwitch(1)
```

Unlock the parameter setting of load.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Pallet Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Instantiate matrix pallet

- Function:

```
Pallet = MatrixPallet (index, "IsUnstack= true Userframe= 1")
```

- Description: instantiate matrix pallet
- Parameter:
 - Index: matrix pallet index
- Optional parameter:
 - IsUnstack: stack mode, range: true or false. true: dismantling mode; false: assembly mode. If not set, the default is assembly mode
 - Userframe: user coordinate system index. If not set, the default is User 0 coordinate system
- Return: matrix pallet object
- Example

```
myPallet = MatrixPallet(0,"IsUnstack=true Userframe=8")
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-09 17:59:44

15.2 Set the next stack index to be operated

- Function:

```
SetPartIndex (Pallet, index)
```

- Description: set the next stack index which is to be operated
- Parameter:
 - Pallet: Pallet object
 - index: the next stack index, initial value: 0
- Return: null
- Example

```
local myPallet = MatrixPallet(0, "IsUnstack=true Userframe=8")  
SetPartIndex(myPallet,1)
```

The next stack index to be operated is 2.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-17 12:41:48

Get the current operated stack index

- Function:

```
GetPartIndex (Pallet)
```

- Description: get the current operated stack index
- Parameter:
 - Pallet: pallet object
- Return: current operated stack index
- Example

```
local index=GetPartIndex(myPallet)
```

If the return value is 1, it indicates that the current operated stack index is 2

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Set the next pallet layer index to be operated

- Function:

```
SetLayerIndex (Pallet, index)
```

- Description: set the next pallet layer index which is to be operated
- Parameter:
 - Pallet: pallet object
 - index: next pallet layer index. initial value: 0
- Return: null
- Example

```
local myPallet = MatrixPallet(0, "IsUnstack=true Userframe=8")  
SetLayerIndex(myPallet,1)
```

The next pallet layer index to be operated is 2.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-17 12:41:48

Get the current pallet layer index

- Function:

```
GetLayerIndex (Pallet)
```

- Description: get the current pallet layer index
- Parameter:
 - Pallet: pallet object
- Return: current pallet layer index
- Example

```
local index=GetLayerIndex(myPallet)
```

If the return value is 1, it indicates that the current operated pallet layer index is 2.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Reset pallet

- Function:

```
Restet (Pallet)
```

- Description: reset pallet
- Parameter:
 - Pallet: pallet object
- Return: null
- Example

```
local myPallet = MatrixPallet(0, "IsUnstack=true Userframe=8")  
Reset(myPallet)
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-17 12:41:48

Check whether the stack assembly or dismantling is completed

- Function:

```
IsDone (Pallet)
```

- Description: check whether the stack assembly or dismantling is completed
- Parameter:
 - Pallet: pallet object
- Return:
 - true: finished
 - false: un-finished
- Example

```
Result = IsDone(myPallet)  
If (result == true)  
...  
...
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Release palletizing instance

- Function:

```
Release (Pallet)
```

- Description: release pallet object
- Parameter:
 - Pallet: pallet object
- Return: null
- Example

```
Release(myPallet)
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

The robot moves from the current position to the first stack position as the configured stack assembly path

- Function:

```
MoveIn (Pallet, "velAB=20 velBC=30 accAB=20 accBC=10 CP=20 SYNC=1")
```

- Description: The robot moves from the current position to the first stack position as the configured stack assembly path
- Required parameter:
 - Pallet: pallet object
- Optional parameter:
 - velAB: velocity rate when the robot moves from the safe point to the prepare point, range: 1-100
 - velBC: velocity rate when the robot moves from the prepare point to the first stack point, range: 1-100
 - accAB: acceleration rate when the robot moves from the safe point to the prepare point, range: 1-100
 - accBC: acceleration rate when the robot moves from the prepare point to the first stack point, range: 1-100
 - CP: whether to set continuous path function, range: 0- 100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Return: null
- Example

```
MoveIn(myPallet, "velAB=90 velBC=50")
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

The robot moves from the current position to the safe point as the configured stack dismantling path

- Function:

```
MoveOut (Pallet, "velAB=20 velBC=30 accAB=20 accBC=10 CP=20 SYNC=1")
```

- Description: the robot moves from the current position to the safe point as the configured stack dismantling path
- Required parameter:
 - Pallet: pallet object
- Optional parameter:
 - velAB: velocity rate when the robot moves from the safe point to the prepare point, range: 1-100
 - velBC: velocity rate when the robot moves from the prepare point to the first stack point, range: 1-100
 - accAB: acceleration rate when the robot moves from the safe point to the prepare point, range: 1-100
 - accBC: acceleration rate when the robot moves from the prepare point to the first stack point, range: 1-100
 - CP: whether to set continuous path function, range: 0- 100
 - SYNC: synchronization flag, range: 0 or 1. If SYNC is 0, it indicates asynchronous execution. This command has a return immediately after being called, regardless of the execution process. If SYNC is 1, it indicates synchronous execution. it will not return after being called until it is executed completely
- Return: null
- Example

```
MoveOut(myPallet, "velAB=90 velBC=50")
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-11 16:37:48

Conveyer Tracking Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Set conveyor number to create a tracing queue

- Function:

```
CnvVison(CnvID)
```

- Description: set conveyor number to create a tracing queue
- Parameter:
 - CnvID: conveyor number
- Return:
 - 0: no error
 - 1: error
- Example

```
CnvVison(1)
```

Send the information (resolution ratio, starting position, direction and bound) of Conveyor 1 to the robot system.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Obtain status of the object

- Function:

```
GetCnvObject(CnvID, ObjID)
```

- Description: obtain the information of the part on the conveyor to check whether the part is in the pickup area
- Parameter:
 - CnvID: conveyor index
 - ObjID: part index
- Return:
 - Part status: whether there is a part, range: true or false
 - Part type
 - Part coordinate (x,y,r)

- Example

```
P111 = {0,0,0}
while true do
  flag,typeObject,P111 = GetCnvObject(0,0)
  if flag == true then
    break
  end
  Sleep(20)
end
```

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Set X-axis, Y-axis offset under the set User coordinate system

- Function:

```
SetCnvPointOffset(xOffset,yOffset)
```

- Description: set X-axis, Y-axis offset under the user coordinate system
- Parameter:
 - xOffset: X axis offset
 - yOffset: Y axis offset unit: mm
- Return:
 - 0: no error
 - 1: error

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Set time compensation

- Function:

```
SetCnvTimeCompensation (time)
```

- Description: set time compensation This command is used for compensating the pick-up position offset in the moving direction of the conveyor which is caused by taking photos with a time delay
- Parameter:
 - time: time-offset, unit: ms
- Return:
 - 0: no error
 - 1: error

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:44:00

Synchronize the specified conveyor

- Function:

```
SyncCnv (CnvID)
```

- Description: synchronize the specified conveyor Only Move command among motion commands between SyncCnv(CnvID) and StopSyncCnv(CnvID) is supported
- Parameter:
 - CnvID: conveyor index
- Return:
 - 0: no error
 - 1: error

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Stop synchronous conveyor

- Function:

```
StopSyncCnv (CnvID)
```

- Description: stop synchronizing the conveyor The other commands following this command will not be executed until this command running is completed
- Parameter:
 - CnvID: conveyor index
- Return:
 - 0: no error
 - 1: error

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Other Commands

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Set the on-off state of safeskin

- Function:

```
SetSafeSkin (status)
```

- Description: set the on-off state of safeskin
- Parameter:
 - Status: on-off state of safeskin, 0: switch off the safeskin; 1: switch on the safeskin
- Return: null
- Example

```
SetSafeSkin (1)
```

Switch on the safeskin.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57

Set the state of obstacle avoidance of safeskin

- Function:

```
SetObstacleAvoid(status)
```

- Description: set the on/off state of obstacle avoidance of safeskin

If the safeskin of Dobot+ is off, this command does not work.

This command only works within the script. The safeskin will return to default "almost-pause" mode once you exit the script.

- Parameter:
 - Status: on-off state of obstacle avoidance of safeskin
 - 0: switch off the obstacle avoidance of safeskin
 - 1: switch on the obstacle avoidance of safeskin
- Return: null
- Example

```
SetObstacleAvoid(status)
```

Switch on the obstacle avoidance of safeskin.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-12 14:09:21

Set collision level

- Function:

```
SetCollisionLevel(level)
```

- Description: set the collision level
- Parameter:
 - level: collision level
 - 0: switch off collision detection
 - 1~5: more sensitive with higher level
- Return: null
- Example

```
SetCollisionLevel(2)
```

Set the collision level as Level 2.

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2021 all right reserved, powered by GitbookRevision:
2021-08-06 18:43:57