

Cobot C# SDK user manual version update content(V3.8.2)

**Revision Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description** |
| 2025/01/08 | V3.7.7 | Create |
| 2025/01/20 | V3.7.8 | 1.Command SetLoadWeight() add loadID param  2.Add Command ComputeToolCoordWithPoints() |
| 2025/06/13 | V3.8.2 | 1. The WeaveSetPara() interface adds the parameter of swing direction roll Angle (deflection around the X-axis of the swing) 2. The WeaveChangeStart() interface adds parameters such as swing number, start speed of welding, and end speed of welding 3. The ExtDevSetUDPComParam() interface adds a parameter for whether to automatically establish a connection after a power-off restart 4. SetCollisionDetectionMethod () interface increases the levels of collision threshold mode selection 5. The PtpFIRPlanningStart() interface is added to unify the extreme values of joint stiffness 6. Add WeldingSetVoltageGradualChangeStart () interface 7. Increase WeldingSetVoltageGradualChangeEnd () interface 8. Increase WeldingSetCurrentGradualChangeStart () interface 9. Add WeldingSetCurrentGradualChangeEnd () interface 10. Add ArcWeldTraceAIChannelCurrent () interface 11. Add ArcWeldTraceAIChannelVoltage () interface 12. Add the ArcWeldTraceCurrentPara() interface 13. Add the ArcWeldTraceVoltagePara() interface 14. Add the GetSmarttoolBtnState() interface 15. Add the ExtAxisGetCoord() interface |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Notes:**

**①This manual is applicable to WebApp version 3.8.2 Cobots and is subject to change without notice. For other versions , please visit：**

[Welcome to FAIR’s documentation! — FAIRINO Collaborative Robot User Manual 3.8.2 documentation](https://fairino-doc-en.readthedocs.io/latest/)

Catalogue

[1 Modify the interface 1](#_Toc18330)

[1.1 Set weave parameters 1](#_Toc22494)

[1.1.1 Interface Description 1](#_Toc17799)

[1.1.2 Example 1](#_Toc8926)

[1.2 Wobble gradient begins 2](#_Toc21204)

[1.2.1 Interface Description 2](#_Toc31284)

[1.2.2 Example 2](#_Toc28861)

[1.3 UDP Extension Axis Communication Parameter Configuration 4](#_Toc20435)

[1.3.1 Interface Description 4](#_Toc4208)

[1.4 Set up the robot collision detection method 4](#_Toc814)

[1.4.1 Interface Description 4](#_Toc25249)

[1.5 Start Ptp motion FIR filtering 5](#_Toc23796)

[1.5.1 Interface Description 5](#_Toc9449)

[2 Added Command 5](#_Toc25969)

[2.1 Selection of AI channels for current feedback in arc tracking 5](#_Toc25361)

[2.1.1 Interface Description 5](#_Toc21301)

[2.2 Selection of AI channels for voltage feedback in arc tracking 5](#_Toc5043)

[2.2.1 Interface Description 5](#_Toc31660)

[2.3 Current feedback conversion parameters of arc tracking 6](#_Toc29396)

[2.3.1 Interface Description 6](#_Toc15827)

[2.4 Voltage feedback Conversion Parameters of Arc Tracking Welding machine 6](#_Toc21094)

[2.4.1 Interface Description 6](#_Toc24361)

[2.4.2 Example 7](#_Toc6106)

[2.5 Set the welding voltage to start gradually 7](#_Toc21898)

[2.5.1 Interface Description 7](#_Toc5919)

[2.6 Set the welding voltage gradient to end 8](#_Toc3234)

[2.6.1 Interface Description 8](#_Toc16069)

[2.7 Set the welding current to start gradually 8](#_Toc27657)

[2.7.1 Interface Description 8](#_Toc26147)

[2.8 Set the welding current gradient to end 9](#_Toc19285)

[2.8.1 Interface Description 9](#_Toc21929)

[2.8.2 Example 9](#_Toc11171)

[2.9 Get the status of the SmartTool button 10](#_Toc8902)

[2.9.1 Interface Description 10](#_Toc2650)

[2.9.2 Example 10](#_Toc1040)

[2.10 Obtain the extended axis coordinate system 11](#_Toc32583)

[2.10.1 Interface Description 11](#_Toc10124)

[2.10.2 Example 12](#_Toc10293)

# Modify the interface

* 1. **Set weave parameters**
     1. **Interface Description**

Section: 13 Robot welding

1. /\*\*
2. \* @brief Set weave parameters
3. \* @param [in] weaveNum parameters number
4. \* @param [in] weaveType weave type：0- plane triangular weave ; 1- vertical L-shaped triangular weave; 2- clockwise circular weave; 3-counterclockwise circular weave; 4-plane sine weave; 5-vertical L-shaped sine weave; 6- vertical triangular weave; 7- Vertical sine weave
5. \* @param [in] weaveFrequency weave frequency(Hz)
6. \* @param [in] weaveIncStayTime Wait mode 0- period does not contain wait time; 1- Period contains the wait time
7. \* @param [in] weaveRange weave amplitude(mm)
8. \* @param [in] weaveLeftRange Vertical triangle swing left chord length (mm)
9. \* @param  [in] weaveRightRange Vertical triangle swing right chord length (mm)
10. \* @param  [in] additionalStayTime Vertical triangle swing vertical triangle point stay time (mm)
11. \* @param [in] weaveLeftStayTime weave left residence time(ms)
12. \* @param [in] weaveRightStayTime weave right residence time(ms)
13. \* @param [in] weaveCircleRadio Circular wiggle-pullback ratio(0-100%)
14. \* @param [in] weaveStationary weave position wait, 0- position continue to move within the waiting time; 1- The position is stationary during the waiting time
15. \* @param [in] weaveYawAngle Swing direction azimuth Angle (rotation around the z-axis of swing), unit °
16. \* @param [in] weaveRotAngle Swing direction lateral Angle (deflection around the X-axis of the swing), unit °
17. \* @return Error code
18. \*/
19. int WeaveSetPara(int weaveNum, int weaveType, double weaveFrequency, int weaveIncStayTime, double weaveRange, double weaveLeftRange, double weaveRightRange, int additionalStayTime, int weaveLeftStayTime, int weaveRightStayTime, int weaveCircleRadio, int weaveStationary, double weaveYawAngle, double weaveRotAngle);
    * 1. **Example**

Section：13.17

1. private void button7\_Click(object sender, EventArgs e)
2. {
3. DescPose startdescPose = new DescPose(146.273, -208.110, 270.102, 177.523, -3.782, -158.101);
4. JointPos startjointPos = new JointPos(98.551, -128.309, 127.341, -87.490, -94.249, -13.208);
5. DescPose enddescPose = new DescPose(146.272, -476.204, 270.102, 177.523, -3.781, -158.101);
6. JointPos endjointPos = new JointPos(93.931, -89.722, 102.216, -101.300, -94.359, -17.840);
7. ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
8. DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);
9. robot.WeaveSetPara(0, 3, 2.000000, 0, 10.000000, 0.000000, 0.000000, 0, 0, 0, 0, 0, 0, 0);
10. robot.MoveL(startjointPos, startdescPose, 2, 0, 100, 100, 100, -1, exaxisPos, 0, 0, offdese);
11. robot.WeaveStart(0);
12. robot.MoveL(endjointPos, enddescPose, 2, 0, 100, 100, 100, -1, exaxisPos, 0, 0, offdese);
13. robot.WeaveEnd(0);
14. robot.WeaveSetPara(0, 3, 2.000000, 0, 10.000000, 0.000000, 0.000000, 0, 0, 0, 0, 0, 0, 30);
15. robot.MoveL(startjointPos, startdescPose, 2, 0, 100, 100, 100, -1, exaxisPos, 0, 0, offdese);
16. robot.WeaveStart(0);
17. robot.MoveL(endjointPos, enddescPose, 2, 0, 100, 100, 100, -1, exaxisPos, 0, 0, offdese);
18. robot.WeaveEnd(0);
19. }
    1. **Wobble gradient begins**
       1. **Interface Description**

Section: 13 Robot welding

1. /\*\*
2. \* @brief Wobble gradient begins
3. \* @param [in] weaveNum Swing number
4. \* @param [in] weaveNum swing number
5. \* @param [in] velStart welding start speed, (cm/min)
6. \* @param [in] velEnd welding end speed, (cm/min)
7. \* @return Error code
8. \*/
9. int WeaveChangeStart(int weaveNum, int weaveNum, double velStart, double velEnd);
   * 1. **Example**

Section：13.17

1. private void button9\_Click(object sender, EventArgs e)
2. {
3. DescPose startdescPose = new DescPose(-319.303, -240.689, 116.379, -175.879, -0.337, 148.239);
4. JointPos startjointPos = new JointPos(20.474, -103.554, 126.774, -116.682, -87.746, -37.709);
5. DescPose enddescPose = new DescPose(-454.166, -327.159, 62.217, 177.199, -2.276, 154.955);
6. JointPos endjointPos = new JointPos(27.176, -74.423, 104.557, -119.315, -93.514, -37.698);
7. DescPose safedescPose = new DescPose(-375.533, -543.319, 19.798, 177.486, -2.489, 175.825);
8. JointPos safejointPos = new JointPos(48.074, -59.714, 89.955, -119.777, -93.508, -37.683);
9. ExaxisPos exaxisPos = new ExaxisPos( 0, 0, 0, 0 );
10. DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0 );
11. robot.WeldingSetCurrentRelation(0, 495, 1, 10, 0);
12. robot.WeldingSetVoltageRelation(10, 45, 1, 10, 1);
13. robot.WeldingSetVoltage(0, 25, 1, 0);// ----设置电压
14. robot.WeldingSetCurrent(0, 260, 0, 0);// ----设置电流
15. robot.MoveJ(safejointPos, safedescPose, 1, 0, 5, 100, 100, exaxisPos, -1, 0, offdese);
16. int rtn=robot.WeldingSetCurrentGradualChangeStart(0, 260, 220, 0, 0);
17. Console.WriteLine($"WeldingSetCurrentGradualChangeStart rtn is {rtn}");
18. rtn=robot.WeldingSetVoltageGradualChangeStart(0, 25, 22, 1, 0);
19. Console.WriteLine($"WeldingSetVoltageGradualChangeStart rtn is {rtn}");
20. rtn = robot.ArcWeldTraceControl(1, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1, 10, 0, 0);
21. Console.WriteLine($"ArcWeldTraceControl rtn is {rtn}");
22. robot.MoveJ(startjointPos, startdescPose, 1, 0, 5, 100, 100, exaxisPos, -1, 0, offdese);
23. robot.ARCStart(0, 0, 10000);
24. robot.WeaveStart(0);
25. rtn=robot.WeaveChangeStart(2, 1, 24, 36);
26. Console.WriteLine($"WeaveChangeStart rtn is {rtn}");
27. robot.MoveL(endjointPos, enddescPose, 1, 0, 100, 100, 2, -1, exaxisPos, 0, 0, offdese);
28. robot.ARCEnd(0, 0, 10000);
29. robot.WeaveChangeEnd();
30. robot.WeaveEnd(0);
31. robot.ArcWeldTraceControl(0, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1, 10, 0, 0);
32. robot.WeldingSetCurrentGradualChangeEnd();
33. robot.WeldingSetVoltageGradualChangeEnd();
34. }
    1. **UDP Extension Axis Communication Parameter Configuration**
       1. **Interface Description**

Section:12 Extension Axis

1. / \* \*
2. \* @brief UDP Extension Axis Communication Parameter Configuration
3. \* @param [in] ip PLC IP address
4. \* @param [in] port number
5. \* @param [in] period Communication cycle (ms, default is 2. Do not modify this parameter)
6. \* @param [in] Loss Packet Time Packet Loss Detection Time (ms)
7. \* @param [in] Packet loss count of kgnum
8. \* @param [in] disconnectTime Communication disconnection confirmation duration
9. \* @param [in] reconnectEnable Communication disconnection Automatic reconnection Enabled 0- Not enabled 1- Enabled
10. \* @param [in] reconnectPeriod (ms)
11. \* @param [in] reconnectNum reconnection times
12. \* @param [in] selfConnect automatically establish a connection upon power-off and restart? 0- No connection established; 1- Establish a connection
13. \* @return error code
14. \*/
15. int ExtDevSetUDPComParam(string ip, int port, int period, int lossPkgTime, int lossPkgNum, int disconnectTime, int reconnectEnable, int reconnectPeriod, int reconnectNum, int selfConnect)
    1. **Set up the robot collision detection method**

Section:6.10

* + 1. **Interface Description**

1. / \*\*
2. \* @brief Set up the robot collision detection method
3. \* @param [in] method Collision detection method: 0- Current mode 1- Dual encoders 2- The current and dual encoders are turned on simultaneously
4. \* @param [in] thresholdMode - Collision level threshold mode; 0-Collision level fixed threshold mode 1- Customize collision detection thresholds
5. \* @return error code
6. \* /
7. int SetCollisionDetectionMethod(int method, int thresholdMode = 0);
   1. **Start Ptp motion FIR filtering**

Section:3.47

* + 1. **Interface Description**

1. / \*\*
2. \* @brief Start Ptp motion FIR filtering
3. \* @param [in] maxAcc Maximum acceleration Extreme Value (deg/s ²)
4. \* @param [in] maxJek Unified Extreme Values of Joint Urgency (deg/s3)
5. \* @return error code
6. \* /
7. int PtpFIRPlanningStart(double maxAcc, double maxJek = 1000);

# Added Command

* 1. **Selection of AI channels for current feedback in arc tracking**
     1. **Interface Description**

Section: 13 Robot welding

1. /\*\*
2. \* @brief Selection of AI channels for current feedback in arc tracking
3. \* @param [in]  channel channels；0-Aux AI0；1-Aux AI1；2-Aux AI2；3-Aux AI3；4-Control Box AI0；5-Control Box AI1
4. \* @return Error code
5. \*/
6. int ArcWeldTraceAIChannelCurrent(int channel);
   1. **Selection of AI channels for voltage feedback in arc tracking**
      1. **Interface Description**

Section: 13 Robot welding

1. /\*\*
2. \* @brief Selection of AI channels for voltage feedback in arc tracking
3. \* @param [in]  channel channels；0-Aux AI0；1-Aux AI1；2-Aux AI2；3-Aux AI3；4-Control Box AI0；5-Control Box AI1
4. \* @return Error code
5. \*/
6. int ArcWeldTraceAIChannelVoltage(int channel);
   1. **Current feedback conversion parameters of arc tracking**
      1. **Interface Description**

Section: 13 Robot welding

1. /\*\*
2. \* @brief Current feedback conversion parameters of arc tracking
3. \* @param [in] AILow AI channel lower limit, default value 0V, range [0-10V]
4. \* @param [in] AIHigh AI channel upper limit, default value 10V, range [0-10V]
5. \* @param [in] The lower limit of the currentLow AI channel corresponds to the current value of the welding machine. The default value is 0V, and the range is [0-200V]
6. \* @param [in] The upper limit of the currentLow AI channel corresponds to the current value of the welding machine. The default value is 0V, and the range is [0-200V]
7. \* @return Error code
8. \*/
9. int ArcWeldTraceCurrentPara(float AILow, float AIHigh, float currentLow, float currentHigh);
   1. **Voltage feedback Conversion Parameters of Arc Tracking Welding machine**
      1. **Interface Description**

Section: 13 Robot welding

1. /\*\*
2. \* @brief Voltage feedback Conversion Parameters of Arc Tracking Welding machine
3. \* @param [in] AILow AI channel lower limit, default value 0V, range [0-10V]
4. \* @param [in] AIHigh AI channel upper limit, default value 10V, range [0-10V]
5. \* @param [in] The lower limit of the voltageLow AI channel corresponds to the welding machine voltage value. The default value is 0V, and the range is [0-200V]
6. \* @param [in] The upper limit of the voltageHigh AI channel corresponds to the voltage value of the welding machine. The default value is 100V, and the range is [0-200V]
7. \* @return Error code
8. \*/
9. int ArcWeldTraceVoltagePara(float AILow, float AIHigh, float voltageLow, float voltageHigh);
   * 1. **Example**
10. private void button8\_Click(object sender, EventArgs e)
11. {
12. DescPose safetydescPose = new DescPose(-504.043, 275.181, 40.908, -28.002, -42.025, -14.044);
13. JointPos safetyjointPos = new JointPos(-39.078, -76.732, 87.227, -99.47, -94.301, 18.714);
14. DescPose startdescPose = new DescPose(-473.86, 257.879, -20.849, -37.317, -42.021, 2.543);
15. JointPos startjointPos = new JointPos(-43.487, -76.526, 95.568, -104.445, -89.356, 3.72);
16. DescPose enddescPose = new DescPose(-499.844, 141.225, 7.72, -34.856, -40.17, 13.13);
17. JointPos endjointPos = new JointPos(-31.305, -82.998, 99.401, -104.426, -89.35, 3.696);
18. ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
19. DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);
20. robot.MoveJ(safetyjointPos, safetydescPose, 1, 0, 20, 100, 100, exaxisPos, -1, 0, offdese);
21. robot.WeldingSetCurrentRelation(0, 495, 1, 10, 0);
22. robot.WeldingSetVoltageRelation(10, 45, 1, 10, 1);
23. robot.WeldingSetVoltage(0, 25, 1, 0);//
24. robot.WeldingSetCurrent(0, 260, 0, 0);//
25. int rtn = robot.ArcWeldTraceAIChannelCurrent(4);
26. Console.WriteLine("ArcWeldTraceAIChannelCurrent rtn is " + rtn);
27. rtn = robot.ArcWeldTraceAIChannelVoltage(5);
28. Console.WriteLine("ArcWeldTraceAIChannelVoltage rtn is " + rtn);
29. rtn = robot.ArcWeldTraceCurrentPara((float)0, (float)5, (float)0, (float)500);
30. Console.WriteLine("ArcWeldTraceCurrentPara rtn is " + rtn);
31. rtn = robot.ArcWeldTraceVoltagePara((float)1.018, (float)10, (float)0, (float)50);
32. Console.WriteLine("ArcWeldTraceVoltagePara rtn is " + rtn);
33. robot.MoveJ(startjointPos, startdescPose, 1, 0, 20, 100, 100, exaxisPos, -1, 0, offdese);
34. robot.ArcWeldTraceControl(1, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1, 10, 0, 0);
35. robot.ARCStart(0, 0, 10000);
36. robot.WeaveStart(0);
37. robot.MoveL(endjointPos, enddescPose, 1, 0, 100, 100, 2, -1, exaxisPos, 0, 0, offdese);
38. robot.ARCEnd(0, 0, 10000);
39. robot.WeaveEnd(0);
40. robot.ArcWeldTraceControl(0, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1, 10, 0, 0);
41. robot.MoveJ(safetyjointPos, safetydescPose, 1, 0, 20, 100, 100, exaxisPos, -1, 0, offdese);
42. }
    1. **Set the welding voltage to start gradually**
       1. **Interface Description**

Section: 13 Robot welding

1. /\*\*
2. \* @brief Set the welding voltage to start gradually
3. \* @param [in] IOType control type; 0- Control Box IO 1- Digital Communication Protocol (UDP) 2- Digital Communication Protocol (ModbusTCP)
4. \* @param [in] voltageStart Initial Welding Voltage (V)
5. \* @param [in] voltageEnd Stop welding Voltage (V)
6. \* @param [in] AOIndex control box AO port number (0-1)
7. \* @param [in] Is blend smooth? 0- Not smooth; 1- Smooth
8. \* @return Error code
9. \*/
10. int WeldingSetVoltageGradualChangeStart(int IOType, double voltageStart, double voltageEnd, int AOIndex, int blend);
    1. **Set the welding voltage gradient to end**
       1. **Interface Description**

Section: 13 Robot welding

1. /\*\*
2. \* @brief Set the welding voltage gradient to end
3. \* @return Error code
4. \*/
5. int WeldingSetVoltageGradualChangeEnd();
   1. **Set the welding current to start gradually**
      1. **Interface Description**

Section: 13 Robot welding

1. /\*\*
2. \* @brief Set the welding current to start gradually
3. \* @param [in] IOType control type; 0- Control Box IO 1- Digital Communication Protocol (UDP) 2- Digital Communication Protocol (ModbusTCP)
4. \* @param [in] voltageStart Initial welding Current (A)
5. \* @param [in] voltageEnd Stop welding current (A)
6. \* @param [in] AOIndex control box AO port number (0-1)
7. \* @param [in] Is blend smooth? 0- Not smooth; 1- Smooth
8. \* @return Error code
9. \*/
10. int WeldingSetCurrentGradualChangeStart(int IOType, double currentStart, double currentEnd, int AOIndex, int blend);
    1. **Set the welding current gradient to end**
       1. **Interface Description**

Section: 13 Robot welding

1. /\*\*
2. \* @brief Set the welding current gradient to end
3. \* @return Error code
4. \*/
5. int WeldingSetCurrentGradualChangeEnd();
   * 1. **Example**
6. private void button9\_Click(object sender, EventArgs e)
7. {
8. DescPose startdescPose = new DescPose(-319.303, -240.689, 116.379, -175.879, -0.337, 148.239);
9. JointPos startjointPos = new JointPos(20.474, -103.554, 126.774, -116.682, -87.746, -37.709);
10. DescPose enddescPose = new DescPose(-454.166, -327.159, 62.217, 177.199, -2.276, 154.955);
11. JointPos endjointPos = new JointPos(27.176, -74.423, 104.557, -119.315, -93.514, -37.698);
12. DescPose safedescPose = new DescPose(-375.533, -543.319, 19.798, 177.486, -2.489, 175.825);
13. JointPos safejointPos = new JointPos(48.074, -59.714, 89.955, -119.777, -93.508, -37.683);
14. ExaxisPos exaxisPos = new ExaxisPos( 0, 0, 0, 0 );
15. DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0 );
16. robot.WeldingSetCurrentRelation(0, 495, 1, 10, 0);
17. robot.WeldingSetVoltageRelation(10, 45, 1, 10, 1);
18. robot.WeldingSetVoltage(0, 25, 1, 0);//
19. robot.WeldingSetCurrent(0, 260, 0, 0);//
20. robot.MoveJ(safejointPos, safedescPose, 1, 0, 5, 100, 100, exaxisPos, -1, 0, offdese);
21. int rtn=robot.WeldingSetCurrentGradualChangeStart(0, 260, 220, 0, 0);
22. Console.WriteLine($"WeldingSetCurrentGradualChangeStart rtn is {rtn}");
23. rtn=robot.WeldingSetVoltageGradualChangeStart(0, 25, 22, 1, 0);
24. Console.WriteLine($"WeldingSetVoltageGradualChangeStart rtn is {rtn}");
25. rtn = robot.ArcWeldTraceControl(1, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1, 10, 0, 0);
26. Console.WriteLine($"ArcWeldTraceControl rtn is {rtn}");
27. robot.MoveJ(startjointPos, startdescPose, 1, 0, 5, 100, 100, exaxisPos, -1, 0, offdese);
28. robot.ARCStart(0, 0, 10000);
29. robot.WeaveStart(0);
30. rtn=robot.WeaveChangeStart(2, 1, 24, 36);
31. Console.WriteLine($"WeaveChangeStart rtn is {rtn}");
32. robot.MoveL(endjointPos, enddescPose, 1, 0, 100, 100, 2, -1, exaxisPos, 0, 0, offdese);
33. robot.ARCEnd(0, 0, 10000);
34. robot.WeaveChangeEnd();
35. robot.WeaveEnd(0);
36. robot.ArcWeldTraceControl(0, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1, 10, 0, 0);
37. robot.WeldingSetCurrentGradualChangeEnd();
38. robot.WeldingSetVoltageGradualChangeEnd();
39. }

## **Get the status of the SmartTool button**

* + 1. **Interface Description**

Section: 10.57

1. / \*\*
2. \* @brief Get the status of the SmartTool button
3. \* @param [out] state SmartTool handle button status; (bit0:0- Communication is normal; 1- Communication disconnection; bit1- Undo operation bit2- Clear the program; bit3-A key bit4-B key bit5-C key bit6-D key bit7-E key bit8-IO key bit9- Manual automatic Starting from bit10
4. \* @return error code
5. \* /
6. int GetSmarttoolBtnState(ref int state);
   * 1. **Example**
7. private void button11\_Click(object sender, EventArgs e)
8. {
9. ROBOT\_STATE\_PKG pkg = new ROBOT\_STATE\_PKG();
10. int state = 0;
11. while (true)
12. {
13. int rtn = robot.GetSmarttoolBtnState(ref state);
14. string binaryString = Convert.ToString(state, 2).PadLeft(32, '0'); // 转换为32位二进制字符串
15. Console.WriteLine($"GetSmarttoolBtnState rtn (binary): {binaryString}");
16. Thread.Sleep(100);
17. }

## **Obtain the extended axis coordinate system**

Section: 12.37；

* + 1. **Interface Description**

1. / \*\*
2. \* @brief Obtain the extended axis coordinate system
3. \* @param [out] coord extended axis coordinate system
4. \* @return error code
5. \* /
6. int ExtAxisGetCoord(ref DescPose coord)
   * 1. **Example**
7. private void button66\_Click(object sender, EventArgs e)
8. {
9. int rtn = robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 200, 1, 100, 5,1);
10. Console.WriteLine("ExtDevSetUDPComParam rtn is " + rtn);
11. string ip = ""; int port = 0; int period = 0; int lossPkgTime = 0; int lossPkgNum = 0; int disconnectTime = 0; int reconnectEnable = 0; int reconnectPeriod = 0; int reconnectNum = 0;
12. rtn = robot.ExtDevGetUDPComParam(ref ip, ref port, ref period, ref lossPkgTime, ref lossPkgNum, ref disconnectTime, ref reconnectEnable, ref reconnectPeriod, ref reconnectNum);
13. string param = "\nip " + ip + "\nport " + port.ToString() + "\nperiod " + period.ToString() + "\nlossPkgTime " + lossPkgTime.ToString() + "\nlossPkgNum " + lossPkgNum.ToString() + "\ndisConntime " + disconnectTime.ToString() + "\nreconnecable " + reconnectEnable.ToString() + "\nreconnperiod " + reconnectPeriod.ToString() + "\nreconnnun " + reconnectNum.ToString();
14. Console.WriteLine("ExtDevGetUDPComParam rtn is " + rtn + param);
15. robot.ExtDevLoadUDPDriver();
16. rtn = robot.ExtAxisServoOn(1, 1);
17. Console.WriteLine("ExtAxisServoOn axis id 1 rtn is " + rtn);
18. rtn = robot.ExtAxisServoOn(2, 1);
19. Console.WriteLine("ExtAxisServoOn axis id 2 rtn is " + rtn);
20. Thread.Sleep(2000);
21. robot.ExtAxisSetHoming(1, 0, 10, 2);
22. Thread.Sleep(2000);
23. rtn = robot.ExtAxisSetHoming(2, 0, 10, 2);
24. Console.WriteLine("ExtAxisSetHoming rtnn is " + rtn);
25. Thread.Sleep(4000);
26. rtn = robot.SetRobotPosToAxis(1);
27. Console.WriteLine("SetRobotPosToAxis rtn is " + rtn);
28. rtn = robot.SetAxisDHParaConfig(1, 128.5f, 206.4f, 0, 0, 0, 0, 0, 0);
29. Console.WriteLine("SetAxisDHParaConfig rtn is " + rtn);
30. rtn = robot.ExtAxisParamConfig(1, 1, 1, 1000, -1000, 1000, 1000, 1.905f, 262144, 200, 1, 0, 0);
31. Console.WriteLine("ExtAxisParamConfig axis 1 rtn is " + rtn);
32. rtn = robot.ExtAxisParamConfig(2, 1, 1, 1000, -1000, 1000, 1000, 4.444f, 262144, 200, 1, 0, 0);
33. Console.WriteLine("ExtAxisParamConfig axis 1 rtn is " + rtn);
34. DescPose toolCoord = new DescPose(0, 0, 210, 0, 0, 0);
35. robot.SetToolCoord(1, toolCoord, 0, 0, 1, 0);
36. JointPos jSafe = new JointPos(115.193f, -96.149f, 92.489f, -87.068f, -89.15f, -83.488f);
37. JointPos j1 = new JointPos(117.559f, -92.624f, 100.329f, -96.909f, -94.057f, -83.488f);
38. JointPos j2 = new JointPos(112.239f, -90.096f, 99.282f, -95.909f, -89.824f, -83.488f);
39. JointPos j3 = new JointPos(110.839f, -83.473f, 93.166f, -89.22f, -90.499f, -83.487f);
40. JointPos j4 = new JointPos(107.935f, -83.572f, 95.424f, -92.873f, -87.933f, -83.488f);
41. DescPose descSafe = new DescPose();
42. DescPose desc1 = new DescPose();
43. DescPose desc2 = new DescPose();
44. DescPose desc3 = new DescPose();
45. DescPose desc4 = new DescPose();
46. ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
47. DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);
48. robot.GetForwardKin( jSafe, ref descSafe);
49. robot.MoveJ( jSafe, descSafe, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
50. Thread.Sleep(2000);
51. robot.GetForwardKin( j1, ref desc1);
52. robot.MoveJ( j1, desc1, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
53. Thread.Sleep(2000);
54. DescPose actualTCPPos = new DescPose();
55. robot.GetActualTCPPose(0, ref actualTCPPos);
56. robot.SetRefPointInExAxisEnd(actualTCPPos);
57. rtn = robot.PositionorSetRefPoint(1);
58. Console.WriteLine("PositionorSetRefPoint 1 rtn is " + rtn);
59. Thread.Sleep(2000);
60. robot.MoveJ( jSafe, descSafe, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
61. robot.ExtAxisStartJog(1, 0, 50, 50, 10);
62. Thread.Sleep(1000);
63. robot.ExtAxisStartJog(2, 0, 50, 50, 10);
64. Thread.Sleep(1000);
65. robot.GetForwardKin( j2, ref desc2);
66. rtn = robot.MoveJ( j2, desc2, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
67. rtn = robot.PositionorSetRefPoint(2);
68. Console.WriteLine("PositionorSetRefPoint 2 rtn is " + rtn);
69. Thread.Sleep(2000);
70. robot.MoveJ( jSafe, descSafe, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
71. robot.ExtAxisStartJog(1, 0, 50, 50, 10);
72. Thread.Sleep(1000);
73. robot.ExtAxisStartJog(2, 0, 50, 50, 10);
74. Thread.Sleep(1000);
75. robot.GetForwardKin( j3, ref desc3);
76. robot.MoveJ( j3, desc3, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
77. rtn = robot.PositionorSetRefPoint(3);
78. Console.WriteLine("PositionorSetRefPoint 3 rtn is " + rtn);
79. Thread.Sleep(2000);
80. robot.MoveJ( jSafe, descSafe, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
81. robot.ExtAxisStartJog(1, 0, 50, 50, 10);
82. Thread.Sleep(1000);
83. robot.ExtAxisStartJog(2, 0, 50, 50, 10);
84. Thread.Sleep(1000);
85. robot.GetForwardKin(j4, ref desc4);
86. robot.MoveJ(j4, desc4, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
87. rtn = robot.PositionorSetRefPoint(4);
88. Console.WriteLine("PositionorSetRefPoint 4 rtn is " + rtn);
89. Thread.Sleep(2000);
90. DescPose axisCoord = new DescPose();
91. robot.PositionorComputeECoordSys(ref axisCoord);
92. robot.MoveJ(jSafe, descSafe, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
93. Console.WriteLine("PositionorComputeECoordSys rtn is {0} {1} {2} {3} {4} {5}", axisCoord.tran.x, axisCoord.tran.y, axisCoord.tran.z, axisCoord.rpy.rx, axisCoord.rpy.ry, axisCoord.rpy.rz);
94. rtn = robot.ExtAxisActiveECoordSys(3, 1, axisCoord, 1);
95. Console.WriteLine("ExtAxisActiveECoordSys rtn is " + rtn);
96. DescPose getCoord=new DescPose(0, 0, 0, 0, 0, 0);
97. rtn = robot.ExtAxisGetCoord(getCoord);
98. }